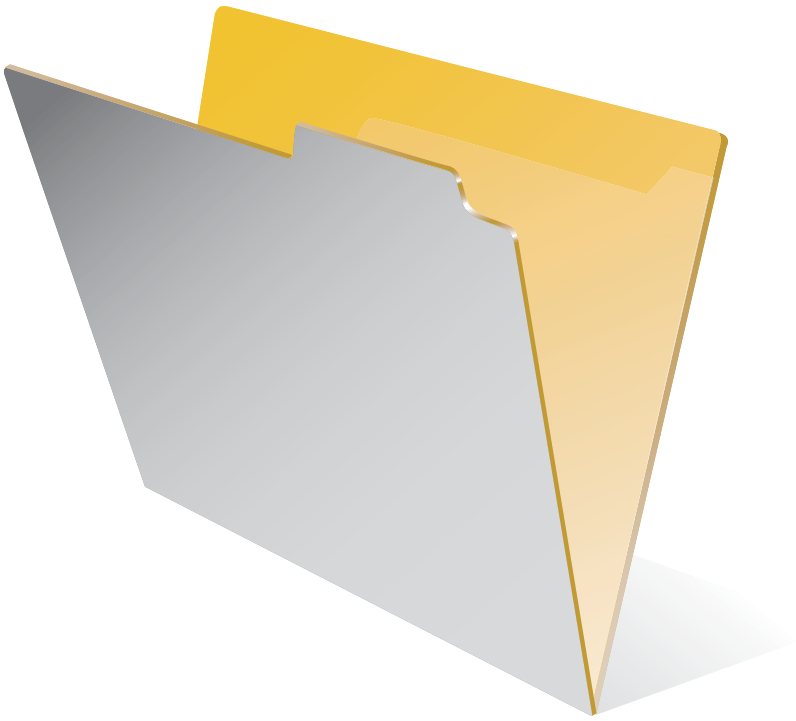
FileMaker® Server 8 Advanced

Custom Web Publishing Handbuch



© 2004-2005 FileMaker, Inc. Alle Rechte vorbehalten.

FileMaker, Inc.
5201 Patrick Henry Drive
Santa Clara, CA 95054, USA

FileMaker ist eine Marke von FileMaker, Inc., eingetragen in den USA und anderen Ländern, und ScriptMaker und das Dateiordner-Logo sind Marken von FileMaker, Inc.

Die FileMaker-Dokumentation ist urheberrechtlich geschützt. Sie dürfen diese Dokumentation ohne schriftliche Genehmigung von FileMaker weder vervielfältigen noch verteilen. Diese Dokumentation darf ausschließlich mit einer gültigen, lizenzierten Kopie der FileMaker-Software verwendet werden.

Alle in den Beispielen erwähnten Personen und Firmen sind rein fiktiv und jegliche Ähnlichkeit mit bestehenden Personen und Firmen ist rein zufällig.

Die Danksagungen und Urheberrechtshinweise finden Sie im entsprechenden Dokument, das mit der Software geliefert wurde.

Weitere Informationen finden Sie auf unserer Website: www.filemaker.de.

Edition: 01

Inhalt

Kapitel 1
Einführung in Custom Web Publishing 11
Über dieses Handbuch 12
Erstellen dynamischer Websites mit der Web Publishing Engine 13
Erläuterung von Custom Web Publishing mit XML 14
Erläuterung von Custom Web Publishing mit XSLT 14
Erläuterung der Tools zur Entwicklung von XSLT-Stylesheets 14
Voraussetzungen für Web Publishing 15
Voraussetzungen für das Veröffentlichen einer Datenbank mit Custom Web Publishing 15
Voraussetzungen für Web-Benutzer zum Zugriff auf eine Custom Web Publishing-Lösung 15
Verbindung zum Internet oder zu einem Intranet 15
Wichtige Funktionen in Custom Web Publishing mit XML und XSLT 16
FileMaker-Scripts und Custom Web Publishing 17
Tipps und Überlegungen zu Scripts 17
Migrieren von Web Publishing-Lösungen aus einer früheren Version von FileMaker Pro 19
Die nächsten Schritte 20

Kapitel 2
Vorbereiten von Datenbanken für Custom Web Publishing 21
Aktivieren von Instant Web Publishing in einer Datenbank 21
Voraussetzungen für Web-Benutzer zum Zugriff auf eine geschützte Datenbank 21
Schützen veröffentlichter Datenbanken 22
Web Server-Unterstützung für Internet-Medientypen (MIME) 23
Veröffentlichen des Inhalts von Medienfeldern im Web 23
Veröffentlichen von Medienfeldobjekten, die in einer Datenbank gespeichert sind 23
Veröffentlichen von Medienfeldobjekten, die als Dateiverweis gespeichert sind 24
Verwendung von Medienfelddaten durch Web-Benutzer 24

Kapitel 3
Zugriff auf XML-Daten mit der Web Publishing Engine 25
Verwenden von Custom Web Publishing mit XML 25
Unterschiede zwischen der Web Publishing Engine und FileMaker Pro XML-Import/Export 25
Wie die Web Publishing Engine aus einer Abfrage XML-Daten generiert 26
Allgemeines Verfahren beim Zugriff auf XML-Daten von der Web Publishing Engine 27

Erläuterung der URL-Syntax für XML-Daten und Medienobjekte 28
Neues in der URL-Syntax für XML-Daten 28
Erläuterung der URL-Syntax für XML-Daten 29
Erläuterung der URL-Syntax für FileMaker-Medienobjekte in XML-Lösungen 30
Erläuterung von URL-Textkodierung 31
Zugriff auf XML-Daten über die Web Publishing Engine 31
Erläuterung von Namensräumen für FileMaker XML 32
Erläuterung der FileMaker-Datenbankfehlercodes 32
Abrufen der Document Type Definitions für die FileMaker-Grammatiken 33
Verwenden der fmresultset-Grammatik 33
Beschreibung der Elemente der fmresultset-Grammatik 33
Beispiel für XML-Daten in der fmresultset-Grammatik 35
Verwenden anderer FileMaker XML-Grammatiken 36
Beschreibung der Elemente in der FMPXMLRESULT Grammatik 37
Beispiel für XML-Daten in der FMPXMLRESULT Grammatik 37
Beschreibung der Elemente der Grammatik FMPXMLLAYOUT 38
XML-Daten in der FMPDSORESULT-Grammatik – Beispiel 39
Erläuterung von UTF-8-kodierten Daten 40
Abfragen von XML-Daten mithilfe von FileMaker-Query-Strings 40
Wechseln des Layouts für eine XML-Antwort 42
Erläuterung der Verarbeitung einer XML-Abfrage 43
Verwenden von Server-seitiger und Client-seitiger Stylesheet-Verarbeitung 43
Fehlerbehebung beim XML-Dokumentzugriff 44

Kapitel 4
Einführung in Custom Web Publishing mit XSLT 45
Erläuterung von FileMaker XSLT-Stylesheets 45
Beispiele für die Verwendung von FileMaker XSLT-Stylesheets 45
Einstieg in die Verwendung von Custom Web Publishing mit XSLT 46
Wie die Web Publishing Engine auf der Basis von XML-Daten und XSLT-Stylesheets Seiten generiert 46
Allgemeine Schritte bei der Verwendung von Custom Web Publishing mit XSLT 47
Generieren von XSLT-Stylesheets mithilfe des FileMaker Site Assistant 48
Installieren des Site Assistant 48
Vor Verwendung des Site Assistant 48
Starten des Site Assistant 49
Verwenden des Site Assistant 49
Erläuterung der vom Site Assistant generierten XSLT-Stylesheets 50

Verwenden von FileMaker CDML Converter 50
Erläuterung von FileMaker CDML Converter 50
Installieren von CDML Converter 51
Starten und Verwenden von CDML Converter 51
Überprüfen und Korrigieren der vom CDML Converter generierten Stylesheets 53
Verwenden der vom CDML Converter generierten Stylesheets 54
Testen der vom CDML Converter generierten Stylesheets 54
Verwenden von FileMaker XSLT-Stylesheets in einer Website oder einem Programm 54
Fehlerbehebung in XSLT-Stylesheets 55

Kapitel 5
Entwickeln von FileMaker XSLT-Stylesheets 57
Verwenden von XSLT-Stylesheets mit der Web Publishing Engine 57
Referenz der FileMaker XSLT-Erweiterungsfunktionen 58
Erläuterung der URL-Syntax für FileMaker XSLT-Stylesheets 58
Erläuterung der URL-Syntax für FileMaker-Medienobjekte in XSLT-Lösungen 59
Verwenden von Query-Strings in FileMaker XSLT-Stylesheets 60
Angeben einer XML-Grammatik für ein FileMaker XSLT-Stylesheet 60
Erläuterung von Namespaces und Präfixen für FileMaker XSLT-Stylesheets 61
Verwenden von statisch definierten Query-Befehlen und Query-Parametern 62
Festlegen von Textkodierung für Abfragen 63
Angeben einer Ausgabemethode und -kodierung 64
Erläuterung der Kodierung von XSLT-Stylesheets 64
Verarbeitung von XSLT-Abfragen, die nicht FileMaker Server abfragen 64
Verwendung von Tokens zur Übergabe von Informationen zwischen Stylesheets 65

Verwenden der FileMaker XSLT-Erweiterungsfunktionen und -parameter 65
Erläuterung der FileMaker-spezifischen XSLT-Parameter, die von der Web Publishing Engine gesetzt werden 66
Verwenden der Query-Information in einer Abfrage 66
Abrufen von Clientinformationen 67
Verwenden des Basis-URI-Parameters der Web Publishing Engine 68
Verwenden des Parameters „authenticated base URI“ 68
Laden zusätzlicher Dokumente 68
Verwenden der Layoutinformation einer Datenbank in einem Stylesheet 69
Verwenden von Inhaltspufferung 70
Verwenden von Web Publishing Engine-Sitzungen zum Speichern von Informationen zwischen Abfragen 71
Verwenden der Sitzungserweiterungsfunktionen 72
Senden von E-Mail-Nachrichten von der Web Publishing Engine 74
Verwenden von Header-Funktionen 76
Verwenden der Cookie-Erweiterungsfunktionen 76
Verwenden der Erweiterungsfunktionen zur Stringmanipulation 78
Vergleich von Strings mithilfe regulärer Perl 5-Ausdrücke 79
Prüfen von Werten in einem Feld, das als Kontrollkästchen formatiert ist 80
Verwenden der Datums-, Zeit- und Tages-Erweiterungsfunktionen 81
Prüfen des Fehlerstatus von Erweiterungsfunktionen 85
Verwenden der Protokollierung 85
Verwenden der Server-seitigen Verarbeitung von Scriptingsprachen 86
Definieren einer Erweiterungsfunktion 86
Ein Beispiel für eine Erweiterungsfunktion 87

Kapitel 6
Testen und Überwachen einer Site 91
Testen einer Custom Web Publishing-Site 91
Beispiele für Stylesheets zum Testen der XML-Ausgabe 91
Überwachen Ihrer Website 92
Verwenden der Web-Server-Zugriffs- und Fehlerprotokolle 92
Verwenden des Web Publishing Engine-Programmprotokolls 93
Verwenden des Web-Server-Modul-Fehlerprotokolls 93
Verwenden von internen Web Publishing Core-Zugriffsprotokollen 94

Anhang A
Gültige Namen in Query-Strings 95
Erläuterung von Query-Befehlen und -Parametern 95
Veraltete Abfragenamen und Parameter 96
Richtlinien für die Verwendung von Query-Befehlen und -Parametern 96
Erläuterung der Referenz von FileMaker-Query-Strings 97
Erläuterung der Syntax für einen voll qualifizierten Feldnamen 97
Hinzufügen von Datensätzen in einem Ausschnitt 98
Bearbeiten von Datensätzen in einem Ausschnitt 99
Erläuterung der Syntax für das Angeben eines Variablenfelds 99
Verwenden der Query-Befehle 100
Query-Befehl –dbnames (Datenbanknamen) 100
Query-Befehl –delete (Datensatz löschen) 100
Query-Befehl –dup (Datensatz duplizieren) 100
Query-Befehl –edit (Datensatz bearbeiten) 101
Query-Befehle –find, –findall oder –findany (Datensätze suchen) 101
–layoutnames (Layoutnamen) (Query-Befehl) 102
–new (Neuer Datensatz) (Query-Befehl) 102
-process (XSLT-Stylesheets verarbeiten) 102
–scriptnames (Scriptnamen) (Query-Befehl) 102
Query-Befehl –view (Layoutinformationen anzeigen) 103

Verwenden der Query-Parameter 103
Query-Parameter –db (Datenbankname) 103
Query-Parameter –encoding (XSLT-Abfrage-Kodierung) 103
Query-Parameter –field (Medienfeldname) 104
Query-Parameter „fieldname“ (Nicht-Medienfeldname) 104
Query-Parameter fieldname.op (Vergleichsoperator) 105
Query-Parameter –grammar (Grammatik für XSLT-Stylesheets) 106
Query-Parameter –lay (Layout) 106
Query-Parameter –lay.response (Layout wechseln für Antwort) 106
Query-Parameter –lop (Logischer Operator) 107
Query-Parameter –max (Maximum Datensätze) 107
Query-Parameter –modid (Änderungs-ID) 107
Query-Parameter –recid (Datensatz-ID) 107
–script (Script) (Query-Parameter) 108
Query-Parameter –script.param (Parameter an Script übergeben) 108
Query-Parameter –script.prefind (Script vor Suchen) 108
Query-Parameter –script.prefind.param (Parameter an Script vor Suchen übergeben) 109
Query-Parameter –script.presort (Script vor Sortieren) 109
Query-Parameter –script.presort.param (Parameter an Script vor Sortieren übergeben) 109
Query-Parameter –skip (Datensätze überspringen) 110
Query-Parameter –sortfield (Sortierfeld) 110
Query-Parameter –sortorder (Sortierfolge) 110
Query-Parameter –stylehref (Stil-Href) 111
Query-Parameter –styletype (Stilart) 112
Query-Parameter –token.[string] (Werte zwischen XSLT-Stylesheets übergeben) 112

Anhang B
Fehlercodes für Custom Web Publishing 113
Fehlercodenummern für FileMaker-Datenbanken 113
Fehlercodenummern für die Web Publishing Engine 120
Fehlercodenummern für die FileMaker XSLT-Erweiterungsfunktionen 121

Anhang C
Konvertieren von CDML-Lösungen in FileMaker XSLT 125

Verfahren bei der Konvertierung von CDML-Lösungen in FileMaker XSLT-Lösungen 125
Konvertierung von CDML-Aktions-Tags, Variablen-Tags und URLs 127
Konvertierung der CDML-Variablen-Tags –error und –errornum 129
Konvertierung veralteter CDML-Aktions-Tags 130
Konvertierung unterstützter CDML-Aktions-Tags 130
Konvertierung veralteter CDML-Variablen-Tags 131
Konvertierung unterstützter CDML-Variablen-Tags 131
Konvertierung von booleschen CDML-Parametern in boolesche XPath-Parameter 133
Konvertierung von booleschen CDML-Operatoren in XPath 134
Konvertierung von CDML-Intratag-Parametern in XSLT-CWP 134
Manuelle Korrektur von CDML-Konvertierungsfehlern 135
Konvertierung von CDML-Ersetzungs-Tags in XSLT-CWP 139

Index 181

Kapitel 1
Einführung in Custom Web Publishing

Mit FileMaker® Server Advanced können Sie FileMaker-Datenbanken im Internet oder in einem Intranet wie folgt veröffentlichen:

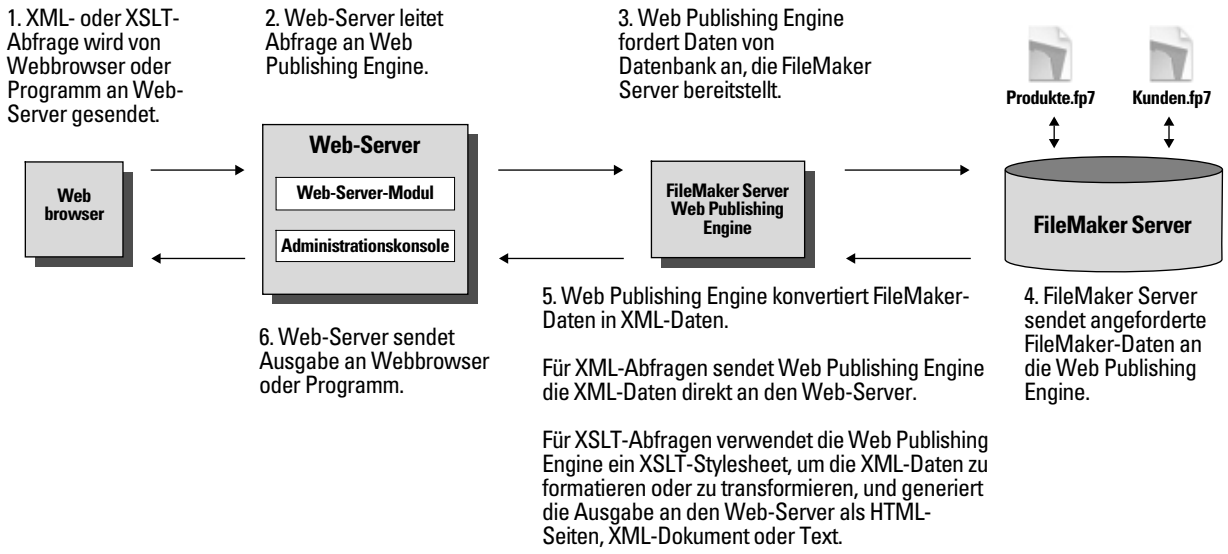
- Custom Web Publishing mit Extensible Markup Language (XML)
- Custom Web Publishing mit XSLT-Stylesheets (Extensible Stylesheet Language Transformations)
- Instant Web Publishing Weitere Informationen finden Sie im *FileMaker Instant Web Publishing Handbuch*.

Custom Web Publishing mit XML und XSLT bietet Ihnen Optionen und Kontrolle hinsichtlich Design und Funktionalität Ihrer Web-Seiten. Für FileMaker Server, der die veröffentlichten Datenbanken bereitstellt, muss weder FileMaker Pro installiert noch Custom Web Publishing verfügbar sein.

Um Instant Web Publishing und Custom Web Publishing mit XML und XSLT zu unterstützen, verwendet FileMaker Server einen Satz von Softwarekomponenten, die so genannte FileMaker Server Web Publishing Engine. Die Web Publishing Engine verwaltet Interaktionen zwischen dem Browser eines Web-Benutzers, Ihrem Web-Server und FileMaker Server. Die Web Publishing Engine fungiert als XSLT-Prozessor und bietet Ausgaben als HTML, XML oder Text (z. B. vCards) auf dem Web-Server, der dann die Ausgabe für den Webbrowser liefert.

Web-Benutzer greifen auf Ihre Custom Web Publishing-Lösung zu, indem sie auf einen HREF-Link klicken oder einen Uniform Resource Locator (URL) eingeben, der die Web-Server-Adresse und eine FileMaker-Query-Zeichenfolgenabfrage angibt. Der URL kann auf XML-Daten zugreifen oder auf ein XSLT-Stylesheet verweisen. Die Web Publishing Engine gibt die XML-Daten, die in der Query-Zeichenfolgenabfrage angegeben sind, oder das Ergebnis des referenzierten XSLT-Stylesheets zurück.

Verwenden der FileMaker Server Web Publishing Engine für Custom Web Publishing mit XML oder XSLT


1. XML- oder XSLT-Abfrage wird von Webbrowser oder Programm an Web-Server gesendet.
2. Web-Server leitet Abfrage an Web Publishing Engine.
3. Web Publishing Engine fordert Daten von Datenbank an, die FileMaker Server bereitstellt.
Produkte.fp7
Kunden.fp7
Web browser
Web-Server
Web-Server-Modul
Administrationskonsole
FileMaker Server Web Publishing Engine
FileMaker Server
6. Web-Server sendet Ausgabe an Webbrowser oder Programm.
5. Web Publishing Engine konvertiert FileMaker-Daten in XML-Daten.
Für XML-Abfragen sendet Web Publishing Engine die XML-Daten direkt an den Web-Server.
Für XSLT-Abfragen verwendet die Web Publishing Engine ein XSLT-Stylesheet, um die XML-Daten zu formatieren oder zu transformieren, und generiert die Ausgabe an den Web-Server als HTML-Seiten, XML-Dokument oder Text.
4. FileMaker Server sendet angeforderte FileMaker-Daten an die Web Publishing Engine.

Wichtig Sicherheitsüberlegungen gewinnen zunehmend an Bedeutung, wenn Sie Daten im Web veröffentlichen. Prüfen Sie die Sicherheitsrichtlinien, die im Handbuch *FileMaker Sicherheitshandbuch*, verfügbar unter www.filemaker.de/downloads, beschrieben sind.

Über dieses Handbuch

Dieses Handbuch setzt voraus, dass Sie mit XML und XSLT, der Entwicklung von Websites und dem Einsatz von FileMaker Pro zur Erstellung von Datenbanken vertraut sind. Dieses Handbuch enthält die folgenden Informationen über Custom Web Publishing mit XML und XSLT auf FileMaker Server:

- Voraussetzungen für die Entwicklung einer Custom Web Publishing-Lösung mit XML oder XSLT
- Veröffentlichen Ihrer Datenbanken mit XML oder XSLT
- Voraussetzungen für Web-Benutzer zum Zugriff auf eine Custom Web Publishing-Lösung
- Abruf von XML-Daten aus Datenbanken, die FileMaker Server bereitstellt
- Entwicklung von FileMaker XSLT-Stylesheets
- Konvertieren von CDML-Lösungen in FileMaker XSLT

Wichtig Sie können PDF-Dateien der FileMaker-Dokumentation von www.filemaker.de/downloads herunterladen. Aktualisierungen dieses Dokuments erhalten Sie ebenfalls auf der Website.

Die Dokumentation für FileMaker Server Advanced umfasst auch die folgende Informationen:

Weitere Informationen über	Siehe
Installieren und Konfigurieren von FileMaker Server	*FileMaker Server Administratorhandbuch* *FileMaker Server Administration Hilfe*
Installieren und Konfigurieren von Instant Web Publishing und Custom Web Publishing mit XML und XSLT für FileMaker Server	*FileMaker Server Advanced Web Publishing Installationshandbuch*

Weitere Informationen über	Siehe
Instant Web Publishing	*FileMaker Instant Publishing Handbuch*
Custom Web Publishing	*FileMaker Server Advanced Custom Web Publishing Handbuch (dieses Handbuch)*
Installation und Konfiguration der ODBC- und JDBC-Treiber	*Installation der FileMaker ODBC- und JDBC-Client-Treiber*
ODBC und JDBC	*FileMaker ODBC- und JDBC-Entwicklerhandbuch*

Erstellen dynamischer Websites mit der Web Publishing Engine

Die Web Publishing Engine bietet Custom Web Publishing für FileMaker Server mithilfe von XML-Datenveröffentlichung und vom Server verarbeiteten XSLT-Stylesheets. Custom Web Publishing bietet mehrere Vorteile:

- **Anpassung**: Sie können festlegen, wie Web-Benutzer mit FileMaker-Daten interagieren und wie die Daten in Webbrowsern angezeigt werden.
- **Datenaustausch**: Mithilfe von FileMaker XML können Sie FileMaker-Daten mit anderen Websites und Programmen austauschen.
- **Datenintegration**: Mit FileMaker XSLT-Stylesheets können Sie beliebige Untergruppen der FileMaker-Daten in andere Websites und mit anderen Middleware- und Spezialprogrammen integrieren. Sie können die Daten so darstellen, als würden sie zu einer anderen Website gehören, anstatt ein vollständiges FileMaker-Layout im Webbrowser anzuzeigen.
- **Sicherheit**: Als Administrator für die Web Publishing Engine können Sie Instant Web Publishing, XML Web Publishing und XSLT Web Publishing einzeln für alle Datenbanken, die der Server bereitstellt, aktivieren oder deaktivieren. Als Eigentümer der FileMaker-Datenbank können Sie den Zugriff der Web-Benutzer auf Instant Web Publishing, XML Web Publishing oder XSLT Web Publishing für jede Datenbank steuern.
- **Server-seitige Stylesheets**: Server-seitige Verarbeitung von XSLT-Stylesheets verhindert unbefugte Anzeige von vertraulicher Datenbankinformation, die mit Client-seitigen Stylesheets möglich sein könnte.
- **Steuern und Filtern von veröffentlichten Daten**: Mit XSLT-Stylesheets können Sie die Daten und die Art der Datenbankinformationen, die Sie veröffentlichen wollen, steuern und filtern und damit unbefugte Nutzung der Datenbank verhindern. Sie können auch Metadaten ausblenden wie z. B. Datenbank- und Feldnamen.
- **Basierend auf offenen Standards**: Ihnen stehen mehr Tools, Ressourcen und Fachpersonal für Custom Web Publishing-Lösungen zur Verfügung. Wenn Sie Standard-XML oder -XSLT kennen, können Sie sofort mit der Entwicklung von Lösungen beginnen, nachdem Sie ein paar spezielle Details über Custom Web Publishing mit XML gelernt haben, z. B. die erforderliche URL-Syntax und Query-Parameter.
- **Unterstützte Migration von CDML-Lösungen**: Das Tool FileMaker CDML Converter hilft Ihnen bei der Konvertierung von Dateien im CDML-Format in XSLT-Stylesheets und ist hilfreich beim Erlernen von XSLT. Dateien im CDML-Format und XSLT-Stylesheets sind ähnlich und die Unterschiede sind leicht verständlich. XSLT-Stylesheets können komplexer und leistungsstärker als CDML-Dateien sein.

Erläuterung von Custom Web Publishing mit XML

Custom Web Publishing mit XML bietet die Möglichkeit, Daten in FileMaker-Datenbanken abzufragen und die Daten bequem auf die gewünschte Weise zu benutzen. Mithilfe einer HTTP-Abfrage mit den geeigneten Query-Befehlen und -Parametern können Sie FileMaker-Daten als ein XML-Dokument abrufen. Sie können die XML-Daten dann in anderen Programmen verwenden oder ihnen ein XSLT-Stylesheet zuweisen Weitere Informationen finden Sie unter Kapitel 3, „Zugriff auf XML-Daten mit der Web Publishing Engine“.

Erläuterung von Custom Web Publishing mit XSLT

Custom Web Publishing mit XSLT ermöglicht das Umwandeln, Filtern oder Formatieren von XML-Daten für die Verwendung in einem Webbrowser oder in anderen Programmen. Sie können ein XSLT-Stylesheet verwenden, um die Daten zwischen einer FileMaker XML-Grammatik und einer anderen XML-Grammatik für den Einsatz in einem anderen Programm oder einer anderen Datenbank umzuwandeln. Sie können die Daten filtern, indem Sie steuern, welche Datenbankfelder vom Stylesheet veröffentlicht werden. Sie können das Erscheinungsbild der Daten in einer Web-Seite formatieren und steuern, wie der Web-Benutzer mit den Daten interagiert. Weitere Informationen finden Sie unter Kapitel 4, „Einführung in Custom Web Publishing mit XSLT“.

Die Web Publishing Engine verwendet Ihre Stylesheets, um Daten dynamisch aus einer FileMaker-Datenbank zu beziehen, wenn ein Web-Benutzer eine HTTP-Abfrage und einen URL sendet, der auf eines Ihrer XSLT-Stylesheets verweist. Die Web Publishing Engine verwendet ein Stylesheet, um die XML-Daten umzuwandeln und zu formatieren, und generiert die resultierende HTML-Seite, mit der der Web-Benutzer arbeiten kann.

Besuchen Sie für zusätzliche Informationen über die Verwendung von FileMaker Server Advanced Custom Web Publishing mit XML und XSLT die Website www.filemaker.de/downloads.

Erläuterung der Tools zur Entwicklung von XSLT-Stylesheets

FileMaker Server Advanced beinhaltet zwei Tools zur Entwicklung von XSLT-Stylesheets:

- Der FileMaker Site Assistant ist eine Anwendung, mit der Sie grundlegende XSLT-Stylesheets als Ausgangspunkt für Custom Web Publishing mit XSLT erstellen können. Der Site Assistant eignet sich ideal, um den Aufbau von FileMaker XSLT-Stylesheets kennen zu lernen. Sie können dann Ihr eigenes Erstellungswerkzeug für XSLT-Stylesheets verwenden, um die Stylesheets je nach Bedarf zu ändern. Weitere Informationen finden Sie unter „Generieren von XSLT-Stylesheets mithilfe des FileMaker Site Assistant“ auf Seite 48.
- FileMaker CDML Converter ist ein Programm, das bestehende Dateien im CDML-Format in XSLT-Stylesheets umwandelt, die mit Custom Web Publishing mit XSLT kompatibel sind. Das ist ein nützliches Tool für den Prozess der CDML-Website-Migration und zum Kennenlernen des Aufbaus von FileMaker XSLT-Stylesheets. Weitere Informationen finden Sie unter „Verwenden von FileMaker CDML Converter“ auf Seite 50.

Voraussetzungen für Web Publishing

Voraussetzungen für das Veröffentlichen einer Datenbank mit Custom Web Publishing

Um Datenbanken mithilfe von Custom Web Publishing mit XML oder XSLT zu veröffentlichen, ist Folgendes erforderlich:

- ein Windows- oder Mac OS X-basierter Computer mit FileMaker Server, der für Custom Web Publishing aktiviert ist
- eine oder mehrere FileMaker Pro-Datenbanken, die FileMaker Server bereitstellt
- die installierte und konfigurierte FileMaker Server Web Publishing Engine
- ein Web-Server, entweder Microsoft IIS unter Windows oder Apache unter Mac OS X
- IP-Adresse oder Domänenname des Hosts, auf dem der Web-Server läuft
- beliebiger Webbrowser und Zugriff auf den Web-Server, um Ihre Custom Web Publishing-Lösung zu entwickeln und zu testen

Voraussetzungen für Web-Benutzer zum Zugriff auf eine Custom Web Publishing-Lösung

Für den Zugriff auf eine Custom Web Publishing-Lösung mit XML oder XSLT benötigen Web-Benutzer:

- eine beliebige Webbrowser-Software
- Zugang zum Internet oder zu einem Intranet und zum Web-Server
- IP-Adresse oder Domänenname des Hosts, auf dem der Web-Server läuft

Wenn die Datenbank passwortgeschützt ist, müssen Web-Benutzer auch einen Benutzernamen und ein Passwort für ein Datenbankkonto eingeben.

Verbindung zum Internet oder zu einem Intranet

Wenn Sie Datenbanken im Internet oder in einem Intranet veröffentlichen, muss auf dem Host-Computer FileMaker Pro oder FileMaker Server laufen und die freizugebenden Datenbanken müssen bereitgestellt und verfügbar sein. Zusätzlich:

- Es wird dringend empfohlen, dass Sie Ihre Datenbank auf einem Computer veröffentlichen, der über eine ständige Verbindung zum Internet bzw. Intranet verfügt. Sie können Datenbanken zwar ohne ständige Verbindung veröffentlichen, aber sie stehen Web-Benutzern nur zur Verfügung, wenn Ihr Computer mit dem Internet bzw. Intranet verbunden ist.
- Der Host-Computer für die Web Publishing Engine sollte über eine eigene *statische* (permanente) IP-Adresse oder einen Domänennamen verfügen. Wenn Sie die Verbindung zum Internet über einen Internet-Anbieter (ISP) herstellen, könnte Ihre IP-Adresse *dynamisch zugewiesen* werden (d. h., sie ändert sich bei jeder Verbindung). Mit einer dynamischen IP-Adresse ist es für Web-Benutzer schwieriger, Ihre Datenbanken zu finden. Wenn Sie sich über Ihre Zugangsart nicht sicher sind, wenden Sie sich an Ihren ISP oder Netzwerkadministrator.

Wichtige Funktionen in Custom Web Publishing mit XML und XSLT

In den Versionen 7 und 8 von FileMaker Server Advanced bietet Custom Web Publishing mit XML und XSLT mehrere wichtige Funktionen:

- Datenbanken werden auf FileMaker Server bereitgestellt und FileMaker Pro muss nicht ausgeführt werden.
- Sie können Server-seitige XSLT-Stylesheet-Verarbeitung nutzen, die mehr Sicherheit als die Client-seitige Stylesheet-Verarbeitung bietet.
- In FileMaker Server 8 Advanced können Sie die Server-seitige Verarbeitung von JavaScript in XSLT-Stylesheets nutzen. Weitere Informationen finden Sie unter „Verwenden der Server-seitigen Verarbeitung von Scriptingsprachen“ auf Seite 86.
- Sie können mit Ihrem FileMaker XSLT-Stylesheet unbefugte Verwendung von Query-Befehlen und Query-Parametern verhindern, indem Sie die zu verwendenden Query-Befehle, Parameter und Werte bei der Anforderung von XML-Daten statisch definieren Weitere Informationen finden Sie unter „Verwenden von statisch definierten Query-Befehlen und Query-Parametern“ auf Seite 62.
- Wie in FileMaker Pro basiert der Zugriff auf Daten, Layouts und Felder auf den Benutzerkontoeinstellungen, die in den Zugriffsrechten der Datenbank definiert sind. Die Web Publishing Engine unterstützt auch mehrere andere Sicherheitsverbesserungen Weitere Informationen finden Sie unter „Schützen veröffentlichter Datenbanken“ auf Seite 22.
- Web-Benutzer können komplexe Scripts mit mehreren Schritten ausführen. Im Web werden jetzt ungefähr 70 Scriptschritte unterstützt. Weitere Informationen finden Sie im nächsten Abschnitt, „FileMaker-Scripts und Custom Web Publishing“.
- In FileMaker Server 8 Advanced können Sie einen Parameterwert an ein FileMaker-Script übergeben. Weitere Informationen finden Sie unter „Query-Parameter –script.param (Parameter an Script übergeben)“ auf Seite 108, „Query-Parameter –script.prefind.param (Parameter an Script vor Suchen übergeben)“ auf Seite 109 und „Query-Parameter –script.presort.param (Parameter an Script vor Sortieren übergeben)“ auf Seite 109.
- Mithilfe der neuen fmresultset-XML-Grammatik können Sie nach Name auf Felder zugreifen und relatedset- (Ausschnitt) Daten manipulieren.
- Mithilfe von Sitzungsfunktionen in einem XSLT-Stylesheet können Sie die Informationen und Transaktionen eines Web-Benutzers in vom Server geführten Sitzungen speichern.
- Einige neue Query-Befehle und -Parameter wurden hinzugefügt, während einige andere veraltet sind. Für den Zugriff auf Daten in einer Datenbank müssen Sie ein Layout angeben. Aus Sicherheitsgründen wird der Datenzugriff ohne Angabe eines Layouts nicht mehr unterstütz. Weitere Informationen finden Sie unter Anhang A, „Gültige Namen in Query-Strings“
- Jeder Web-Benutzer kann über einen eindeutigen Variablenfeldwert verfügen, der für die Dauer der Sitzung bestehen bleibt. Allgemeine Informationen zu Variablenfeldern finden Sie in der FileMaker Pro Hilfe. Informationen über die Verwendung von Variablenfeldern mit Custom Web Publishing finden Sie unter „Erläuterung der Syntax für das Angeben eines Variablenfelds“ auf Seite 99.

FileMaker-Scripts und Custom Web Publishing

Die Funktion ScriptMaker™ in FileMaker Pro ist nützlich, um sich häufig wiederholende Aufgaben zu automatisieren oder mehrere Aufgaben zusammenzufassen. In Kombination mit Custom Web Publishing ermöglichen FileMaker-Scripts den Web-Benutzern, mehrere Aufgaben oder eine Reihe von Aufgaben durchzuführen.

FileMaker unterstützt ungefähr 70 Scriptschritte in Custom Web Publishing. Web-Benutzer können eine Vielzahl automatisierter Aufgaben ausführen, wenn Sie Scripts in einer Query-Zeichenfolge für einen URL oder in einer <?xslt–cwp–query?>-Verarbeitungsanweisung in einem XSLT-Stylesheet verwenden. Um nicht unterstützte Scriptschritte anzuzeigen, wählen Sie im Dialogfeld „Script bearbeiten“ in FileMaker Pro die Option Web-Kompatibilität anzeigen. Grau dargestellte Scripts werden nicht im Web unterstützt. Informationen über das Erstellen von Scripts finden Sie in der FileMaker Pro Hilfe.

Tipps und Überlegungen zu Scripts

Auch wenn viele Scriptschritte im Web identisch funktionieren, gibt es einige, die sich wie in der folgenden Tabelle beschrieben anders verhalten. Testen Sie alle Scripts, die von einem Webbrowser ausgeführt werden, bevor Sie Ihre Datenbank bereitstellen. Stellen Sie sicher, dass Sie sich mit unterschiedlichen Benutzerkonten anmelden, damit Sie sichergehen können, dass die Scripts für alle Clients gleichermaßen funktionieren.

Beachten Sie folgende Tipps und Überlegungen:

- Verwenden Sie Konten und Zugriffsrechte, um die Scripts einzuschränken, die ein Web-Benutzer ausführen kann. Überprüfen Sie, ob die Scripts nur Web-kompatible Scriptschritte enthalten und nur Zugang zu Scripts gewähren, die von einem Webbrowser aus ausgeführt werden sollen.
- Berücksichtigen Sie die Nebeneffekte von Scripts, die eine Reihe von Scriptschritten ausführen, die durch Zugriffsrechte kontrolliert werden. Wenn ein Script beispielsweise einen Scriptschritt zum Löschen von Datensätzen enthält und der Web-Benutzer sich nicht mit einem Konto anmeldet, das das Löschen von Datensätzen zulässt, führt das Script den Scriptschritt zum Löschen von Datensätzen nicht aus. Das Script könnte jedoch weiter ausgeführt werden, so dass unerwartete Ergebnisse auftreten können.
- Für Scripts wählen Sie Script mit vollen Zugriffsrechten ausführen, damit Scripts Aufgaben ausführen können, für die Sie dem Benutzer keinen Zugang gewähren wollen. Beispielsweise können Sie Benutzern untersagen, mit ihren Konten und Zugriffsrechten Datensätze zu löschen, ihnen aber die Ausführung eines Scripts gestatten, das unter vordefinierten Bedingungen innerhalb eines Scripts bestimmte Arten von Datensätzen löscht.
- Wenn Ihre Scripts Schritte enthalten, die nicht unterstützt werden (z. B. Schritte, die nicht Web-kompatibel sind), verwenden Sie den Scriptschritt AnwenderAbbruchZulassen setzen, um festzulegen, wie nachfolgende Schritte gehandhabt werden.
  - Wenn der Scriptschritt Anwender Abbruch Zulassen setzen aktiviert (ein) ist, hindern nicht unterstützte Scriptschritte das Script an der Fortsetzung.
  - Wenn AnwenderAbbruchZulassen setzen ausgeschaltet ist, werden nicht unterstützte Scriptschritte übersprungen und das Script setzt die Ausführung fort.
  - Wenn dieser Scriptschritt nicht enthalten ist, werden Scripts ausgeführt, als ob die Funktion aktiv wäre, d. h., nicht unterstützte Scriptschritte stoppen Scripts.

- Einige Scripts, die mit einem Schritt von einem FileMaker Pro-Client aus funktionieren, benötigen eventuell einen zusätzlichen Schritt „Datensatz/Abfrage bestätigen“, um die Daten auf dem Host zu speichern. Da Web-Benutzer nicht über eine direkte Verbindung zum Host verfügen, werden sie nicht sofort benachrichtigt, wenn sich Daten ändern. Funktionen wie bedingte Wertelisten funktionieren z. B. nicht gleichermaßen für Web-Benutzer, da die Daten auf dem Host gespeichert werden müssen, bevor sich die Wirkung in einem Wertelistenfeld zeigt.
- In ähnlicher Weise sollten alle Scripts, die Daten verändern, den Schritt „Schreibe Änderung Datens./Abfrage“ enthalten, da alle Datenänderungen erst im Browser sichtbar werden, wenn die Daten gespeichert bzw. am Server bestätigt wurden. Dies gilt für einige Scriptschritte wie Ausschneiden, Kopieren, Einfügen usw. Viele Aktionen mit nur einem Scriptschritt sollten in Scripts umgewandelt werden, die den Schritt „Schreibe Änderung Datens./Abfrage“ enthalten. Wenn Sie Scripts entwerfen, die von einem Webbrowser aus ausgeführt werden, empfiehlt es sich, den Schritt „Schreibe Änderung Datens./Abfrage“ am Ende des Scripts aufzunehmen, um sicherzustellen, dass alle Änderungen gespeichert werden.
- Um von der Art des Clients abhängige Scripts zu erstellen, verwenden Sie die Funktion Hole(ProgrammVersion). Wenn der zurückgegebene Wert „Web Publishing Engine 8.0v1“ enthält, wissen Sie, dass der aktuelle Benutzer über Custom Web Publishing auf Ihre Datenbank zugreift. Weitere Informationen zu Funktionen finden Sie in der FileMaker Pro Hilfe.
- Nachdem Sie Ihre Dateien konvertiert haben, sollten Sie jedes Script öffnen, das Web-Benutzer ausführen könnten, und die Option Web-Kompatibilität anzeigen aktivieren, um sicherzustellen, dass das Script mit Custom Web Publishing einwandfrei ausgeführt wird.
- Wenn Sie ein Script in einem XSLT-Stylesheet benutzen, das einen Status festlegt oder ändert, müssen Sie mithilfe der Administrationskonsole die Option XSLT-Datenbanksitzungen für die Web Publishing Engine aktivieren. Andernfalls werden Status zwischen Abfragen nicht beibehalten. Weitere Informationen erhalten Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.

Die nachfolgenden Scriptschritte arbeiten im Web anders als in FileMaker Pro. Weitere Informationen zu allen Scriptschritten finden Sie in der FileMaker Pro Hilfe.

Scriptschritt	Verhalten in Custom Web Publishing-Lösungen
Script ausführen	Scripts können nur dann in anderen Dateien ausgeführt werden, wenn die Dateien auf FileMaker Server bereitgestellt werden und Custom Web Publishing in den anderen Dateien aktiviert ist.
Programm beenden	Meldet Web-Benutzer ab, schließt alle Fenster, beendet aber den Webbrowser nicht.
AnwenderAbbruchZulassen setzen	Legt fest, wie nicht unterstützte Scriptschritte gehandhabt werden. Aktivieren, damit Scripts gestoppt werden; deaktivieren, damit nicht unterstützte Schritte übersprungen werden. Weitere Informationen finden Sie weiter oben. **Hinweis** Web-Benutzer können Custom Web Publishing-Scripts nicht abbrechen, aber diese Option ermöglicht, dass nicht unterstützte Scriptschritte die Fortsetzung des Scripts stoppen.
Fehleraufzeichnung setzen	Diese Option ist bei Custom Web Publishing immer aktiviert. Web-Benutzer können Custom Web Publishing-Scripts nicht abbrechen.

Scriptschritt	Verhalten in Custom Web Publishing-Lösungen
Scriptpause setzen	Dieses Script wird zwar in Custom Web Publishing unterstützt, sollte aber nicht benutzt werden. Wenn der Scriptschritt „Scriptpause setzen“ ausgeführt wird, hält das Script an. Nur ein Script, das erneut den Schritt „Scriptpause setzen“ enthält, kann die Ausführung fortsetzen. Wenn die Scriptausführung im Pausezustand bleibt, bis die Sitzung endet, wird das Script nicht vollständig ausgeführt.
Datensätze sortieren	Eine festgelegte Sortierfolge muss mit dem Scriptschritt gespeichert werden, um mit Custom Web Publishing ausgeführt zu werden.
URL öffnen	Dieses Script hat keine Wirkung in einer Custom Web Publishing-Lösung.
Gehe zu Feld	Im Webbrowser können Sie „Gehe zu Feld“ nicht verwenden, um zu einem bestimmten Feld zu wechseln. Sie können diesen Scriptschritt jedoch in Kombination mit anderen Scriptschritten verwenden, um Aufgaben auszuführen. Beispielsweise können Sie zu einem Feld gehen und den Inhalt kopieren und dann zu einem anderen Feld gehen und den Wert einfügen. Um die Wirkung im Browser zu sehen, müssen Sie den Datensatz mit dem Scriptschritt „Datensatz bestätigen“ speichern.
Schreibe Änderung Datens./Abfrage	Sendet den Datensatz zur Datenbank.

Migrieren von Web Publishing-Lösungen aus einer früheren Version von FileMaker Pro

Lesen Sie, bevor Sie bestehende Datenbanken in FileMaker Pro 8 aktualisieren, das Handbuch *Konvertieren von FileMaker-Datenbanken aus früheren Versionen*, das unter www.filemaker.de/downloads verfügbar ist.

Beginnen Sie die Migration der Web Publishing-Lösung, indem Sie Ihre Dateien sichern und eine separate Testumgebung für die Konvertierung einrichten. Testen Sie nach der Entwicklung der XML- oder XSLT-Lösung ihre Funktionalität und Sicherheit (Konten und Berechtigungen), bevor Sie die Site freigeben.

Einige weitere Überlegungen bei der Migration von Lösungen, die Sie mit XML oder XSLT veröffentlichen wollen, speziell für Lösungen, die vor Version 7 erstellt wurden:

- Das Zugriffsrechtemodell wurde in FileMaker Pro 7 verbessert. Ziehen Sie in Erwägung, Benutzernamen und Passwörter neu zu vergeben, um Konten und Berechtigungen zu nutzen. Weitere Informationen finden Sie in der FileMaker Pro Hilfe.
- Plugins für die Verwendung mit FileMaker Pro werden nicht automatisch für FileMaker Server aktiviert. Weitere Informationen erhalten Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.
- Die Web-Sicherheitsdatenbank wird nicht mehr unterstützt. Wenn Sie sie für Web-basierte Sicherheit verwendet haben, müssen Sie die Konten, Passwörter und entsprechenden Berechtigungen auf Ihre konvertierten Datenbankdateien in FileMaker Pro übertragen (siehe das Handbuch *Konvertieren von FileMaker-Datenbanken aus früheren Versionen*).
- Wenn Sie eine CDML-Lösung migrieren, ziehen sie die Informationen in Anhang C, „Konvertieren von CDML-Lösungen in FileMaker XSLT“ zu Rate.

- Falls ein Datenbank-Medienfeld einen Dateiverweis anstelle eines tatsächlichen Objekts speichert, muss das entsprechende Medienobjekt im Ordner FileMaker Pro Web gespeichert sein, wenn der Datensatz erstellt oder bearbeitet wird, und dann in einen Ordner mit dem gleichen relativen Speicherort im Root-Ordner der Web-Server-Software kopiert oder verschoben werden. Weitere Informationen finden Sie unter „Veröffentlichen des Inhalts von Medienfeldern im Web“ auf Seite 23.
- Die URL-Syntax für Web-veröffentlichte Dateien hat sich in FileMaker Pro Server 7 Advanced geändert. Wenn Sie Links für den Zugriff auf eine Datenbank im Web mit Versionen vor FileMaker Pro Server 7 Advanced erstellt haben, müssen Sie die Links mit der neuen Syntax und dem neuen Datenbankspeicherort aktualisieren, nachdem die Datei konvertiert und auf FileMaker Server bereitgestellt wurde. Weitere Informationen hierzu finden Sie unter „Erläuterung der URL-Syntax für XML-Daten und Medienobjekte“ auf Seite 28 und „Erläuterung der URL-Syntax für FileMaker XSLT-Stylesheets“ auf Seite 58.
- Die Scriptunterstützung für das Web wurde in FileMaker Pro 7 verbessert.
  - Zugriffsrechte für Web-Benutzer können so eingestellt werden, dass bestimmte einzelne Scripts ausgeführt werden können und andere nicht ausführbar sind.
  - Das ScriptMaker-Dialogfeld „Script bearbeiten“ enthält die Option Web-Kompatibilität anzeigen, um anzuzeigen, ob ein Scriptschritt „Web-kompatibel“ ist. Für Web-Benutzer können spezielle Scripts erstellt werden. Nachdem Sie Ihre Dateien konvertiert haben, öffnen Sie jedes Script, das Web-Benutzer ausführen könnten, und aktivieren Sie Web-Kompatibilität anzeigen, um festzustellen, ob Ihre Scripts Schritte enthalten, die zu unerwarteten Ergebnissen führen könnten, wenn sie von einem Webbrowser aus ausgeführt werden.
  - Im Web werden Scripts immer mit der Option Fehleraufzeichnung setzen Ein ausgeführt.

Die nächsten Schritte

Einige Vorschläge für Aktionen und weitere Informationen in der Dokumentation, mit deren Hilfe Sie den Einstieg in die Entwicklung von Custom Web Publishing-Lösungen finden:

- Falls noch nicht geschehen, aktivieren Sie Custom Web Publishing mithilfe von FileMaker Server Administration und der Web Publishing Engine Administrationskonsole (siehe die FileMaker Server Administration Hilfe und das Handbuch *FileMaker Server Advanced Web Publishing Installation*).
- Öffnen Sie in FileMaker Pro jede FileMaker-Datenbank, die Sie veröffentlichen wollen, und stellen Sie sicher, dass in der Datenbank die geeigneten erweiterten Zugriffsrechte für Custom Web Publishing aktiviert sind Weitere Informationen finden Sie unter „Aktivieren von Instant Web Publishing in einer Datenbank“ auf Seite 21.
- Wie Sie mithilfe von XML auf Daten in FileMaker-Datenbanken zugreifen, erfahren Sie in Kapitel 3, „Zugriff auf XML-Daten mit der Web Publishing Engine“.
- Wie Sie mit der Entwicklung von FileMaker XSLT-Stylesheets beginnen, lesen Sie in Kapitel 4, „Einführung in Custom Web Publishing mit XSLT“.

Kapitel 2
Vorbereiten von Datenbanken für Custom Web Publishing

Bevor Sie Custom Web Publishing mit einer Datenbank verwenden können, müssen Sie die Datenbank vorbereiten und vor unbefugtem Zugriff schützen.

Aktivieren von Instant Web Publishing in einer Datenbank

Sie müssen Custom Web Publishing in jeder Datenbank aktivieren, die Sie veröffentlichen wollen. Sie können Custom Web Publishing mit XML oder Custom Web Publishing mit XSLT in jeder Datenbank einzeln oder beide Technologien gleichzeitig aktivieren. Wenn Sie in einer Datenbank keine dieser Technologien aktivieren, können Web-Benutzer Custom Web Publishing nicht für den Zugriff auf die Datenbank verwenden, selbst wenn sie von FileMaker Server bereitgestellt wird, der zur Unterstützung einer Web Publishing Engine konfiguriert ist.

So aktivieren Sie Custom Web Publishing in einer Datenbank:

1. Öffnen Sie in FileMaker Pro die zu veröffentlichende Datenbank mit einem Konto, das über die Berechtigung für vollen Zugriff verfügt. Alternativ können Sie die Datenbank mit einem Konto öffnen, das über die Berechtigung „Erweiterte Zugriffsrechte verwalten“ verfügt.
2. Weisen Sie einer oder mehreren Berechtigungen eine oder beide dieser erweiterten Zugriffsrechte zu:
   - Verwenden Sie für Custom Web Publishing mit XML dieses Schlüsselwort: fmxml
   - Verwenden Sie für Custom Web Publishing mit XSLT dieses Schlüsselwort: fmxslt

   In FileMaker Pro Version 8 sind die Schlüsselwörter fmxml und fmxslt im Register „Erweiterte Zugriffsrechte“ für Sie definiert.
3. Weisen Sie die Berechtigung(en), die die erweiterten Custom Web Publishing-Zugriffsrechte enthalten, einem oder mehreren Konten bzw. dem Admin- oder Gastkonto zu.

Hinweis Wenn Sie Kontonamen und Passwörter für Custom Web Publishing-Lösungen definieren, verwenden Sie druckbare ASCII-Zeichen. Beispiel: a-z, A-Z und 0-9. Verwenden Sie für sicherere Kontonamen und Passwörter zudem Satzzeichen wie „!“ und „%“, aber verwenden Sie keine Doppelpunkte. Informationen über das Einrichten von Konten finden Sie in der FileMaker Pro Hilfe.

Voraussetzungen für Web-Benutzer zum Zugriff auf eine geschützte Datenbank

Wenn Web-Benutzer eine Custom Web Publishing-Lösung für den Zugriff auf eine Datenbank verwenden, werden sie eventuell aufgefordert, ihre Kontoinformationen einzugeben. Wenn das Gastkonto für die Datenbank deaktiviert ist oder über keine Berechtigung mit dem erweiterten Zugriffsrecht für Custom Web Publishing verfügt, verwendet die Web Publishing Engine die HTTP-Standardauthentifizierung, um entsprechende Informationen von Web-Benutzern zu verlangen. Der Browser des Web-Benutzers zeigt das Dialogfeld für die HTTP-Standardauthentifizierung an, in dem der Benutzer einen Benutzernamen und ein Passwort für ein Konto eingibt, das über ein erweitertes Zugriffsrecht für Custom Web Publishing verfügt.

Zusammenfassung der Abläufe, wenn ein Web-Benutzer eine Custom Web Publishing-Lösung für den Zugriff auf eine Datenbank verwendet:

- Wenn Sie einem Konto kein Passwort zugewiesen haben, geben Web-Benutzer nur den Kontonamen an.
- Wenn das Gastkonto deaktiviert ist, werden die Benutzer beim Zugriff auf die Datenbank aufgefordert, Kontoname und Passwort anzugeben. Für das Konto muss ein erweitertes Zugriffsrecht für Custom Web Publishing aktiviert sein.
- Wenn das Gastkonto aktiviert ist und über eine Berechtigung mit einem erweiterten Zugriffsrecht für Custom Web Publishing verfügt, öffnen automatisch alle Web-Benutzer die Datenbank mit den Zugriffsrechten, die dem Gastkonto zugewiesen sind. Wenn das erweiterte Zugriffsrecht für Custom Web Publishing dem Gastkonto zugewiesen ist:
  - Web-Benutzer werden nicht aufgefordert, Kontoname und Passwort einzugeben, wenn sie eine Datei öffnen.
  - Alle Web-Benutzer werden automatisch mit dem Gastkonto angemeldet und erhalten die Gast-Zugriffsrechte. Sie können Benutzern über den Scriptschritt „Erneut anmelden“ erlauben, ihre Anmeldekonten von einem Webbrowser aus zu wechseln (z. B. vom Gastkonto zu einem anderen Konto mit mehr Rechten).
  - Die Standardberechtigungen für Gastkonten umfassen den Nur-Lese-Zugriff. Sie können für dieses Konto die Standardberechtigungen einschließlich erweiterten Zugriffsrechten ändern. Weitere Informationen finden Sie in der FileMaker Pro Hilfe.
- Web-Benutzer können ihr Kontopasswort allgemein nicht von einem Webbrowser aus ändern. Es ist jedoch möglich, diese Funktion über den Scriptschritt „Passwort ändern“ in Ihre Datenbank zu integrieren (damit Web-Benutzer ihr Passwort ändern können). Weitere Informationen finden Sie in der FileMaker Pro Hilfe.

Schützen veröffentlichter Datenbanken

Wenn Sie Custom Web Publishing mit XML oder XSLT verwenden, können Sie beschränken, wer auf Ihre veröffentlichten Datenbanken zugreifen kann.

- Weisen Sie Datenbankkonten, die für Custom Web Publishing verwendet werden, Passwörter zu.
- Aktivieren Sie Custom Web Publishing mit XML oder XSLT nur in den Berechtigungen für Konten, denen Sie den Zugriff auf Ihre veröffentlichten Datenbanken erlauben wollen.
- Um einen Typ der Custom Web Publishing-Technologie für eine einzelne Datenbank zu aktivieren oder zu deaktivieren, wählen Sie das entsprechende erweiterte Zugriffsrecht.
- Um einen Typ der Custom Web Publishing-Technologie für alle Custom Web Publishing-Lösungen in der Web Publishing Engine zu aktivieren oder zu deaktivieren, verwenden Sie die Administrationskonsole. Weitere Informationen finden Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.

- Sie können Ihren Web-Server konfigurieren, um die IP-Adressen einzuschränken, die über die Web Publishing Engine auf Ihre Datenbanken zugreifen können. Beispielsweise können Sie angeben, dass nur Web-Benutzer von der IP-Adresse 192.168.100.101 auf Ihre Datenbanken zugreifen können. Informationen über das Einschränken von IP-Adressen finden Sie in der Dokumentation zu Ihrem Web-Server.
- Sie können Secure Sockets Layer (SSL)-Verschlüsselung für die Kommunikation zwischen Ihrem Web-Server und den Browsern von Web-Benutzern verwenden. Die SSL-Verschlüsselung wandelt Informationen, die zwischen Servern und Clients übertragen werden, mithilfe von mathematischen Formeln in unverständliche Informationen um. Der englische Fachausdruck für diese Chiffrier-Algorithmen ist Ciphers. Diese Algorithmen nutzt der Empfänger, um mithilfe von Schlüsseln, den so genannten Chiffrierschlüsseln, die Informationen wieder in verständliche Daten umzuwandeln. Informationen über das Aktivieren und Konfigurieren von SSL erhalten Sie in der Dokumentation zu Ihrem Web-Server.

Weitere Informationen über die Sicherheit Ihrer veröffentlichten Datenbanken finden Sie im *FileMaker Sicherheitshandbuch*.

Web Server-Unterstützung für Internet-Medientypen (MIME)

Ihr Web-Server bestimmt die Unterstützung für die aktuellen MIME-Typen (MIME; Multipurpose Internet Mail Extensions), die für das Internet registriert sind. Die Web Publishing Engine ändert die MIME-Unterstützung eines Web-Servers nicht. Weitere Informationen finden Sie in der Dokumentation zu Ihrem Web-Server.

Veröffentlichen des Inhalts von Medienfeldern im Web

Der Inhalt eines Medienfelds, z. B. eine Bilddatei, kann entweder in einer FileMaker-Datenbank oder als Dateiverweis mit einem relativen Pfad gespeichert werden.

Hinweis Die Web Publishing Engine unterstützt kein Streaming von Filmen. Web-Benutzer müssen eine ganze Filmdatei herunterladen, bevor sie den Film ansehen können.

Veröffentlichen von Medienfeldobjekten, die in einer Datenbank gespeichert sind

Wenn ein Medienfeld die tatsächlichen Dateien in der FileMaker-Datenbank speichert, ist keine Aktion mit dem Inhalt des Medienfelds erforderlich, sofern die Datenbankdatei korrekt auf dem FileMaker Server freigegeben und zugänglich ist. (Weitere Informationen hierzu finden Sie unter „Erläuterung der URL-Syntax für FileMaker-Medienobjekte in XML-Lösungen“ auf Seite 30, und „Erläuterung der URL-Syntax für FileMaker-Medienobjekte in XSLT-Lösungen“ auf Seite 59.)

Veröffentlichen von Medienfeldobjekten, die als Dateiverweis gespeichert sind

Wenn ein Medienfeld Dateiverweise anstelle tatsächlicher Dateien speichert, müssen Sie folgende Schritte ausführen, um die Medienfeldobjekte mit der Web Publishing Engine zu veröffentlichen.

Hinweis Alle QuickTime-Filme werden als Verweis in einem Medienfeld gespeichert.

So veröffentlichen Sie Medienfeldobjekte, die als Dateiverweise gespeichert sind:

1. Speichern Sie die Medienobjektdateien im Web-Ordner des FileMaker Pro-Ordners.
2. Fügen Sie in FileMaker Pro die Objekte in das Medienfeld ein und wählen Sie die Option Nur Verweis auf die Datei speichern.
3. Kopieren oder verschieben Sie die Objektdateien, auf die verwiesen wird, im Web-Ordner an die gleiche Stelle im relativen Pfad des Root-Ordners der Web-Server-Software.
   - IIS: Verschieben Sie die Dateien in: <*Root-Laufwerk*>\Inetpub\wwwroot
   - Apache: Verschieben Sie die Dateien in: /Library/WebServer/Documents

Hinweis Für Medienobjekte, die als Dateiverweise gespeichert sind, muss Ihr Web-Server so konfiguriert sein, dass er die MIME-Typen für alle Arten von Dateien unterstützt, die Sie anbieten wollen (z. B. Filme). Weitere Informationen finden Sie in der Dokumentation zu Ihrem Web-Server.

Verwendung von Medienfelddaten durch Web-Benutzer

Wenn Sie eine Datenbank mit der Web Publishing Engine im Web veröffentlichen, können Web-Benutzer wie folgt eingeschränkt mit Daten in Medienfeldern arbeiten:

- Web-Benutzer können keine Töne abspielen oder OLE-Objekte in einem Medienfeld anzeigen – stattdessen wird eine Grafik angezeigt.
- Web-Benutzer können den Inhalt eines Medienfelds nicht ändern oder ihm Inhalte hinzufügen.
- Falls Ihre Datenbank Grafiken enthält, die nicht in GIF- oder JPEG-Format sind, erstellt die Web Publishing Engine ein temporäres JPEG-Bild, wenn die Grafikdaten von einem Webbrowser angefordert werden.

Kapitel 3
Zugriff auf XML-Daten mit der Web Publishing Engine

Sie können FileMaker-Daten in Extensible Markup Language- (XML) Format mithilfe der Web Publishing Engine beziehen und aktualisieren. So wie HTML zur Standardbeschreibungssprache für die Kommunikation im World Wide Web wurde, hat sich XML zur Standardsprache für den strukturierten Datenaustausch entwickelt. Diese Sprache wird von zahlreichen Einzelpersonen, Organisationen und Unternehmen zur Übermittlung von Produktinformationen, Transaktionen, Inventardaten und anderen geschäftlichen Informationen genutzt.

Verwenden von Custom Web Publishing mit XML

Wenn Sie Standard-XML kennen, können Sie sofort mit dem Einsatz der Web Publishing Engine beginnen, nachdem Sie ein paar spezielle Details über Custom Web Publishing mit XML gelernt haben, z. B. die erforderliche URL-Syntax und Query-Parameter.

Indem Sie HTTP URL-Abfragen mit FileMaker-spezifischen Query-Befehlen und -Parametern verwenden, können Sie eine Datenbank abfragen, die von FileMaker Server bereitgestellt wird, und die resultierenden Daten in XML-Format herunterladen. Sie können eine Datenbank beispielsweise auf alle Datensätze in einem bestimmten Postleitzahlenbereich abfragen und die resultierenden XML-Daten wie gewünscht verwenden.

Sie können auch die Server-seitigen XSLT-Stylesheets der Web Publishing Engine verwenden, um die XML-Daten zu filtern, die Daten in HTML oder Text wie vCards umzuformatieren oder die Daten in andere XML-Grammatiken wie Scalable Vector Graphics (SVG) umzuwandeln. (Weitere Informationen hierzu finden Sie unter Kapitel 4, „Einführung in Custom Web Publishing mit XSLT“ und Kapitel 5, „Entwickeln von FileMaker XSLT-Stylesheets“).

Allgemeine Informationen zu XML, weitere Beispiele, in denen es verwendet wird, sowie Links zu XML-Ressourcen finden Sie auf der FileMaker-Website unter www.filemaker.de.

Hinweis Die von der Web Publishing Engine generierten XML-Daten sind wohlgeformt und richten sich nach der XML 1.0-Spezifikation. Informationen zu den Anforderungen des XML-Aufbaus finden Sie in der XML-Spezifikation unter www.w3.org.

Unterschiede zwischen der Web Publishing Engine und FileMaker Pro XML-Import/Export

Die Web Publishing Engine und FileMaker Pro gestatten Ihnen, XML-Daten mit FileMaker-Datenbanken zu verwenden. Es gibt jedoch einige wesentliche Unterschiede zwischen den beiden Methoden:

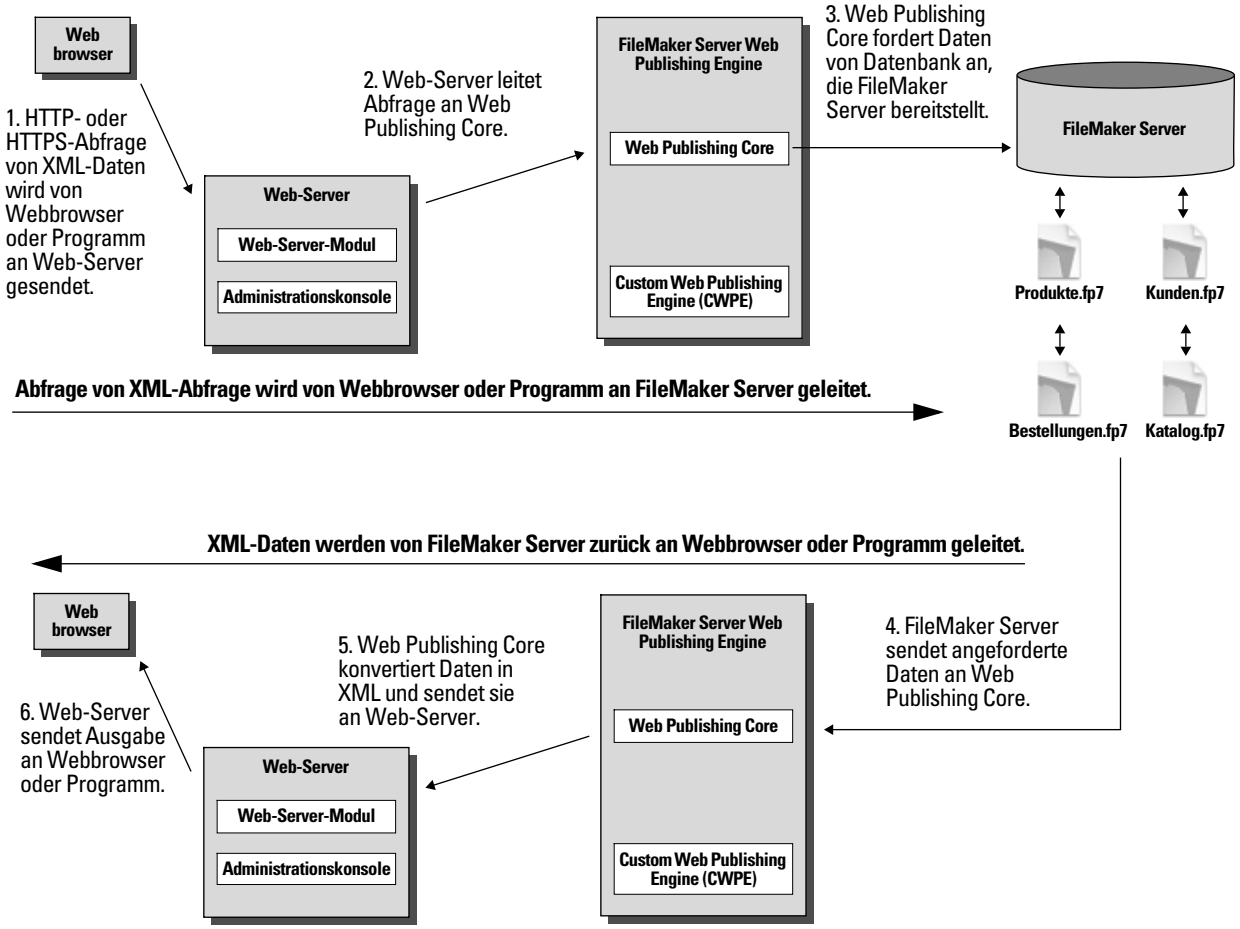
- Für den Zugriff auf XML-Daten und XSLT-Web Publishing unterstützt die Web Publishing Engine die fmresultset-, die FMPXMLRESULT- und die FMPXMLLAYOUT-Grammatik. Für XML-Import verwendet FileMaker Pro die FMPXMLRESULT-Grammatik und für den Export die FMPXMLRESULT- oder FMPDSORESULT-Grammatik. Weitere Informationen finden Sie unter „Zugriff auf XML-Daten über die Web Publishing Engine“ auf Seite 31.
- Um mit der Web Publishing Engine auf XML-Daten zuzugreifen, verwenden Sie einen Web Publishing Engine-Query-String in einem URL. Für den Import und Export von XML mit FileMaker Pro verwenden Sie FileMaker Pro-Menübefehle oder Scripts.

- Die Web Publishing Engine ist serverbasiert und kann auf demselben oder einem anderen Host als FileMaker Server installiert werden. FileMaker Pro XML-Import und -Export ist desktopbasiert.
- Sie können dynamisch auf XML-Daten von FileMaker-Datenbanken zugreifen, indem Sie URL-Abfragen mit der Web Publishing Engine verwenden. Die XML-Exportfunktion von FileMaker Pro erzeugt eine zuvor angegebene XML-Datendatei.
- Das Arbeiten mit XML-Daten über die Web Publishing Engine ist ein interaktiver Vorgang. FileMaker Pro XML-Import und -Export erfolgt im Stapelbetrieb.
- Die Web Publishing Engine kann auf XML-Daten von einem FileMaker-Ausschnitt zugreifen, FileMaker Pro nicht.
- Die Web Publishing Engine kann auf Daten in einem Medienfeld zugreifen, FileMaker Pro nicht.
- Die Web Publishing Engine bietet Echtzeitzugriff auf FileMaker-Daten über HTTP oder HTTPS, FileMaker Pro nicht.

Hinweis Informationen über das Verwenden von FileMaker Pro für Import und Export von Daten in XML-Format finden Sie in der FileMaker Pro Hilfe.

Wie die Web Publishing Engine aus einer Abfrage XML-Daten generiert

Nachdem eine Abfrage von XML-Daten an den Web-Server gesendet wurde, fragt die Web Publishing Engine die FileMaker-Datenbank ab und gibt die Daten als XML-Dokument zurück.


Web browser
1. HTTP- oder HTTPS-Abfrage von XML-Daten wird von Webbrowser oder Programm an Web-Server gesendet.
2. Web-Server leitet Abfrage an Web Publishing Core.
Web-Server
Web-Server-Modul
Administrationskonsole
FileMaker Server Web Publishing Engine
Web Publishing Core
Custom Web Publishing Engine (CWPE)
3. Web Publishing Core fordert Daten von Datenbank an, die FileMaker Server bereitstellt.
FileMaker Server
Produkte.fp7
Kunden.fp7
Bestellungen.fp7
Katalog.fp7
Abfrage von XML-Abfrage wird von Webbrowser oder Programm an FileMaker Server geleitet.
XML-Daten werden von FileMaker Server zurück an Webbrowser oder Programm geleitet.
Web browser
6. Web-Server sendet Ausgabe an Webbrowser oder Programm.
5. Web Publishing Core konvertiert Daten in XML und sendet sie an Web-Server.
Web-Server
Web-Server-Modul
Administrationskonsole
FileMaker Server Web Publishing Engine
Web Publishing Core
Custom Web Publishing Engine (CWPE)
4. FileMaker Server sendet angeforderte Daten an Web Publishing Core.

Allgemeines Verfahren beim Zugriff auf XML-Daten von der Web Publishing Engine

Im Folgenden erhalten Sie einen Überblick über das Verfahren beim Verwenden der Web Publishing Engine für den Zugriff auf XML-Daten in einer FileMaker-Datenbank:

1. Stellen Sie in der Web Publishing Engine-Administrationskonsole sicher, dass „XML Publishing“ aktiviert ist. Weitere Informationen finden Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.
2. Öffnen Sie in FileMaker Pro jede FileMaker-Datenbank, die Sie veröffentlichen wollen, und stellen Sie sicher, dass in der Datenbank das erweiterte Zugriffsrecht fmxml für XML Custom Web Publishing aktiviert ist. Weitere Informationen finden Sie unter „Aktivieren von Instant Web Publishing in einer Datenbank“ auf Seite 21.

   Um auf XML-Daten in einem Ausschnitt zuzugreifen, stellen Sie die Ansicht für das Datenbanklayout auf Als Formular anzeigen oder Als Liste anzeigen ein. Wenn ein Benutzer oder Script die Ansicht des Datenbanklayouts zu Als Tabelle anzeigen ändert, kann nur auf den ersten Bezugsdatensatz (erste Zeile im Ausschnitt) als XML-Daten zugegriffen werden.

   Die XML-Daten werden in einer Reihenfolge ausgegeben, die der Reihenfolge entspricht, in der die Feldobjekte dem Layout hinzugefügt wurden. Wenn die XML-Datenreihenfolge der Reihenfolge entsprechen soll, in der die Felder auf dem Bildschirm angezeigt werden (von oben nach unten, von links nach rechts), dann wählen Sie alle Felder aus, gruppieren Sie sie und heben Sie dann die Gruppierung wieder auf. Dieser Vorgang setzt die Layoutreihenfolge wieder zurück auf die Bildschirmreihenfolge.
3. Senden Sie durch ein HTML-Formular, einen HREF-Link oder ein Script in Ihrem Programm oder Ihrer Web-Seite eine HTTP- oder HTTPS-Abfrage in Form eines URLs mit Angabe der FileMaker XML-Grammatik, einen Query-Befehl und einen oder mehrere FileMaker-Query-Parameter an die Web Publishing Engine. Der URL kann auch in einem Browser eingegeben werden.

   Informationen über die Angabe des URLs finden Sie im nächsten Abschnitt, „Erläuterung der URL-Syntax für XML-Daten und Medienobjekte“. Informationen über Query-Befehle und -Parameter finden Sie unter „Abfragen von XML-Daten mithilfe von FileMaker-Query-Strings“ auf Seite 40, und in Anhang A, „Gültige Namen in Query-Strings“.
4. Die Web Publishing Engine verwendet die im URL angegebene Grammatik, um XML-Daten mit den Ergebnissen Ihrer Abfrage zu erzeugen (z. B. eine Gruppe von Datensätzen aus der Datenbank), und gibt diese an Ihr Programm oder Ihren Webbrowser zurück.
5. Der Webbrowser, falls er über einen XML-Parser verfügt, zeigt die Daten an bzw. das Programm verwendet die Daten wie von Ihnen angegeben.

   Wenn Sie ein Client-seitiges Stylesheet angegeben haben, weist der Parser des Webbrowsers auch die Stylesheet-Anweisungen zu. Weitere Informationen finden Sie unter „Verwenden von Server-seitiger und Client-seitiger Stylesheet-Verarbeitung“ auf Seite 43.

Erläuterung der URL-Syntax für XML-Daten und Medienobjekte

Dieser Abschnitt beschreibt die URL-Syntax beim Verwenden der Web Publishing Engine für den Zugriff auf XML-Daten und Medienobjekte in FileMaker-Datenbanken. Die URL-Syntax für die Verwendung von XSLT-Stylesheets unterscheidet sich von XML. (Weitere Informationen hierzu finden Sie unter „Erläuterung der URL-Syntax für FileMaker XSLT-Stylesheets“ auf Seite 58 und „Erläuterung der URL-Syntax für FileMaker-Medienobjekte in XSLT-Lösungen“ auf Seite 59.)

Neues in der URL-Syntax für XML-Daten

In den Versionen 7 und 8 von FileMaker Server verwendet die Web Publishing Engine für den Zugriff auf XML-Daten eine neue URL-Syntax.

- FileMaker Pro 6 und frühere Versionen verwendeten in XML-Datenabfragen die folgende Syntax:

  FMPro?<CGI_request>

  Diese Syntax hat sich geändert. Weitere Informationen finden Sie im nächsten Abschnitt, „Erläuterung der URL-Syntax für XML-Daten“.

- Der Query-Parameter –lay für die Angabe eines Datenbanklayouts ist bei allen Query-Befehlen erforderlich mit Ausnahme von –dbnames, –layoutnames, –scriptnames, und –process (nur XSLT-Abfragen). Weitere Informationen finden Sie unter Anhang A, „Gültige Namen in Query-Strings“.

- Der Parameter –format ist veraltet. Die XML-Grammatik für XML-Abfragen wird in der URL-Syntax vor dem Query-String angegeben. Weitere Informationen finden Sie im nächsten Abschnitt, „Erläuterung der URL-Syntax für XML-Daten“.

Hinweis Im Unterschied zu XML-Abfragen wird die Grammatik für XSLT-Stylesheets mit dem Query-Parameter –grammar angegeben. Weitere Informationen finden Sie unter „Angeben einer XML-Grammatik für ein FileMaker XSLT-Stylesheet“ auf Seite 60.

Erläuterung der URL-Syntax für XML-Daten

Die URL-Syntax beim Verwenden der Web Publishing Engine für den Zugriff auf XML-Daten in FileMaker-Datenbanken lautet:

<scheme>://<host>[:<port>]/fmi/xml/<xml_grammar>.xml[?<query string>]

Dabei gilt:

- <scheme> kann das HTTP- oder HTTPS-Protokoll sein.
- <host> ist die IP-Adresse oder der Domänenname des Hosts, auf dem der Web-Server installiert ist.
- <port> ist optional und gibt den Port an, den der Web-Server überwacht. Ohne Portangabe wird der Standard-Port des Protokolls (Port 80 für HTTP oder Port 443 für HTTPS) verwendet.
- <xml_grammar> ist der Name der FileMaker-XML-Grammatik. Mögliche Werte sind fmresultset.xml, FMPXMLRESULT.xml, FMPXMLLAYOUT.xml oder FMPDSORESULT.xml. Weitere Informationen finden Sie unter „Verwenden der fmresultset-Grammatik“ auf Seite 33 und „Verwenden anderer FileMaker XML-Grammatiken“ auf Seite 36.
- <query string> ist eine Kombination aus einem Query-Befehl und einem oder mehreren Query-Parametern für FileMaker XML. (Der Befehl –dbnames verlangt keine Parameter.) Weitere Informationen hierzu finden Sie unter „Abfragen von XML-Daten mithilfe von FileMaker-Query-Strings“ auf Seite 40, und Anhang A, „Gültige Namen in Query-Strings“.

Hinweis Die URL-Syntax einschließlich der Namen von Query-Befehl und -Parametern unterscheidet Groß- und Kleinschreibung, mit Ausnahme von Teilen des Query-Strings. Der URL ist vorwiegend in Kleinbuchstaben mit Ausnahme der drei Grammatiknamen in Großbuchstaben: FMPXMLRESULT, FMPXMLLAYOUT und FMPDSORESULT. Informationen über die Regeln für Groß-/Kleinschreibung im Query-String finden Sie unter „Richtlinien für die Verwendung von Query-Befehlen und -Parametern“ auf Seite 96.

Zwei URL-Beispiele für den Zugriff auf XML-Daten über die Web Publishing Engine:

http://server.company.com/fmi/xml/fmresultset.xml?–db=products&–lay=sales&–findall

http://192.168.123.101/fmi/xml/FMPXMLRESULT.xml?–db=products&–lay=sales&–findall

Erläuterung der URL-Syntax für FileMaker-Medienobjekte in XML-Lösungen

In einem generierten XML-Dokument für eine XML-Lösung unterscheidet sich die Syntax für den Verweis auf ein Medienobjekt für Medienfelder, die das tatsächliche Objekt in der Datenbank speichern, und für Medienfelder, die einen Verweis auf das Objekt speichern.

- Wenn ein Medienfeld das tatsächliche Objekt in der Datenbank speichert, verwendet das <data>-Element des Medienfelds die folgende relative URL-Syntax für den Verweis auf das Objekt:

  <data>/fmi/xml/cnt/data.<extension>?<query string></data>

  Dabei gilt: <extension> ist die Dateinamenerweiterung zur Identifizierung des Objekttyps wie beispielsweise .jpg. Die Dateinamenerweiterung stellt den MIME-Typ ein, damit der Webbrowser die Mediendaten korrekt identifizieren kann. Informationen über <query string> finden Sie im vorherigen Abschnitt, „Erläuterung der URL-Syntax für XML-Daten“.

  Beispiel:

  <data>/fmi/xml/cnt/data.jpg?–db=products&–lay=sales&–field=product_image(1)&–recid=2</data>

  Hinweis In der generierten XML für ein Medienfeld ist der Wert für den Query-Parameter –field ein vollständig qualifizierter Feldname. Die Zahl in Klammern gibt die Wiederholungsnummer für das Medienfeld an und wird für Wiederhol- und für Einzelfelder generiert. Weitere Informationen finden Sie unter „Erläuterung der Syntax für einen voll qualifizierten Feldnamen“ auf Seite 97.

  Verwenden Sie zum Abrufen der Mediendaten aus der Datenbank die folgende Syntax:

  <scheme>://<host>[:<port>]/fmi/xml/cnt/data.<extension>?<query string>

  Informationen über <schema>, <host> oder <port> finden Sie im vorherigen Abschnitt, „Erläuterung der URL-Syntax für XML-Daten“.

  Beispiel:

  http://www.company.com/fmi/xml/cnt/data.jpg?–db=products&–lay=sales&–field=product_image(1)&–recid=2

- Wenn ein Medienfeld einen Dateiverweis anstelle eines tatsächlichen Objekts speichert, enthält das <data>-Element des Medienfelds einen relativen Pfad, der auf das Objekt verweist. Beispiel:

  <data>/images/logo.jpg</data>

  Hinweis Das Medienobjekt, auf das verwiesen wird, muss im Ordner FileMaker Pro Web gespeichert sein, wenn der Datensatz erstellt oder bearbeitet wird, und dann in einen Ordner mit dem gleichen relativen Speicherort im Root-Ordner der Web-Server-Software kopiert oder verschoben werden. Weitere Informationen finden Sie unter „Veröffentlichen des Inhalts von Medienfeldern im Web“ auf Seite 23.

- Für ein leeres Medienfeld ist auch das <data>-Element leer.

Hinweis Die Syntax für Medienobjekte mit XML unterscheidet sich von der Syntax für Medienobjekte mit XSLT. Weitere Informationen finden Sie unter „Erläuterung der URL-Syntax für FileMaker-Medienobjekte in XSLT-Lösungen“ auf Seite 59.

Erläuterung von URL-Textkodierung

Die URLs für den Zugriff auf XML-Daten und Medienfelder müssen in UTF-8 (Unicode Transformation 8 Bit)-Format kodiert werden. Informationen finden Sie unter „Erläuterung von UTF-8-kodierten Daten“ auf Seite 40.

Um z. B. den Wert des Info-Felds in Matinée zu ändern, könnten Sie folgenden URL verwenden:

http://server.company.com/fmi/xml/fmresultset.xml?–db=members&–lay=relationships&–recid=2
&info= fianc%C3%A9e&–edit

In diesem Beispiel-URL ist %C3%A9 die URL-kodierte UTF-8-Darstellung des Zeichens é.

Weitere Informationen zur URL-Textkodierung finden Sie in der URL-Spezifikation unter www.w3.org.

Zugriff auf XML-Daten über die Web Publishing Engine

Um über die Web Publishing Engine auf XML-Daten zuzugreifen, verwenden Sie einen URL, der den Namen der zu verwendenden FileMaker-Grammatik, einen FileMaker-Query-Befehl und einen oder mehrere FileMaker-Query-Parameter angibt. Die Web Publishing Engine generiert XML-Daten aus Ihrer Datenbank, die nach einem der folgenden Typen von XML-Grammatiken formatiert werden:

- **fmresultset:** Dies ist die empfohlene Grammatik für die Web Publishing Engine. Sie ist flexibel und für die Erstellung von XSLT-Stylesheets mit bequemerem Feldzugriff nach Name und einfacherer Änderung von relatedset- (Ausschnitt-) Daten optimiert. Diese Grammatik ist auch direkter mit FileMaker-Terminologie verbunden und bietet z. B. globale Speicheroptionen und Identifikation von Statistik- und Formelfeldern. Sie können diese Grammatik für den Zugriff auf XML-Daten und für XSLT-Stylesheets verwenden. Für einfacheres Web Publishing wurde diese Grammatik knapper formuliert als die FMPXMLRESULT-Grammatik. Weitere Informationen finden Sie unter „Verwenden der fmresultset-Grammatik“ auf Seite 33.
- **FMPXMLRESULT und FMPXMLLAYOUT:** Sie können auch die FMPXMLRESULT- und FMPXMLLAYOUT-Grammatiken mit der Web Publishing Engine für den Zugriff auf XML-Daten und für XSLT-Stylesheets verwenden. Um ein einziges Stylesheet sowohl für XML-Export als auch für Custom Web Publishing einzusetzen, müssen Sie die FMPXMLRESULT-Grammatik benutzen. Für den Zugriff auf Wertelisten und Feldanzeigedaten in Layouts müssen Sie die FMPXMLLAYOUT-Grammatik verwenden. Weitere Informationen finden Sie unter „Verwenden anderer FileMaker XML-Grammatiken“ auf Seite 36.
- **FMPDSORESULT:** Die FMPDSORESULT-Grammatik, die in FileMaker Pro für den Export von XML unterstützt wird, steht für den Zugriff auf XML-Daten über die Web Publishing Engine nicht zur Verfügung. Die FMPDSORESULT-Grammatik wird für XSLT-Stylesheets nicht unterstützt. Informationen über die FMPDSORESULT-Grammatik finden Sie in der FileMaker Pro Hilfe.

Abhängig von der Grammatik, die Sie in der URL-Abfrage angegeben haben, generiert die Web Publishing Engine mit einer der folgenden Grammatiken ein XML-Dokument. Jedes XML-Dokument enthält eine XML-Standard-Namespace-Deklaration für die Grammatik. Informationen finden Sie im nächsten Abschnitt, „Erläuterung von Namensräumen für FileMaker XML“. Verwenden Sie eine dieser Grammatiken in Ihrem Dokument oder auf Ihrer Web-Seite, um FileMaker-Daten im XML-Format anzuzeigen und zu bearbeiten.

Hinweis Von der Web Publishing Engine generierte XML-Daten werden im UTF-8-Format (Unicode Transformation Format 8) kodiert. Weitere Informationen finden Sie unter „Erläuterung von UTF-8-kodierten Daten“ auf Seite 40.

Erläuterung von Namensräumen für FileMaker XML

Mithilfe eindeutiger XML-Namensräume können Sie XML-Tags der richtigen Anwendung zuordnen. Wenn Ihr XML-Dokument beispielsweise zwei <DATABASE>-Elemente enthält, eines für FileMaker Pro-XML-Daten, das andere für Oracle-XML-Daten, kann anhand der Namensräume festgestellt werden, welches <DATABASE>-Element zu welchen XML-Daten gehört.

Die Web Publishing Engine generiert für jede Grammatik einen Standard-Namespace.

Grammatik	Generierter Standard-Namespace
fmresultset	xmlns="http://www.filemaker.com/xml/fmresultset"
FMPXMLRESULT	xmlns="http://www.filemaker.com/ fmpxmlresult"
FMPXMLLAYOUT	xmlns="http://www.filemaker.com/fmpxmllayout"

Erläuterung der FileMaker-Datenbankfehlercodes

Die Web Publishing Engine gibt in den „error code“-Elementen am Beginn jedes XML-Dokuments einen Fehlercode für einen etwaigen Fehler zurück, der im zuletzt ausgeführten Query-Befehl aufgetreten ist. Null (0) bedeutet keinen Fehler.

Grammatik	Verwendete Syntax
fmresultset	<error code="0"></error>
FMPXMLRESULT	<ERRORCODE>0</ERRORCODE>
FMPDSORESULT	<ERRORCODE>0</ERRORCODE>

Das „error code“-Element im XML-Dokument gibt Fehler an, die die Datenbank und Query-Strings betreffen. Andere Fehlertypen können für XSLT-Stylesheets auftreten und werden anders behandelt. Weitere Informationen finden Sie unter Anhang B, „Fehlercodes für Custom Web Publishing“.

Abrufen der Document Type Definitions für die FileMaker-Grammatiken

Sie können die Document Type Definitions (DTDs) für die FileMaker-Grammatiken mithilfe einer HTTP-Abfrage abrufen.

Grammatik	HTTP-Abfrage
fmresultset	http://<host>[:<Port]/fmi/xml/fmresultset.dtd
FMPXMLRESULT	http://<Host>[:<Port]/fmi/xml/FMPXMLRESULT.dtd
FMPXMLLAYOUT	http://<Host>[:<Port]/fmi/xml/FMPXMLLAYOUT.dtd
FMPDSORESULT	http://<Host>[:<Port]/fmi/xml/FMPDSORESULT.dtd?–db=<Datenbank>&–lay=<Layout>

Verwenden der fmresultset-Grammatik

Die XML-Elementnamen in dieser Grammatik verwenden FileMaker-Terminologie und die Speicherung von Feldern erfolgt getrennt von den Feldtypen. Die Grammatik umfasst auch die Fähigkeit, Statistik-, Formel- und Variablenfelder zu identifizieren.

Geben Sie zur Verwendung der fmresultset-Grammatik den folgenden Namen der fmresultset-Grammatik im URL an, der das XML-Dokument von der Web Publishing Engine anfordert:

fmresultset.xml

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=family&–findall

Hinweis Geben Sie die fmresultset-Grammatik unbedingt in Kleinbuchstaben an.

Die Web Publishing Engine generiert ein XML-Dokument mithilfe der fmresultset-Grammatik. Im XML-Dokument verweist die Web Publishing Engine auf die Document Type Definition für die fmresultset-Grammatik in der Anweisung <!DOCTYPE> in der zweiten Zeile des Dokuments, unmittelbar nach der Anweisung <?xml...?>. Die Anweisung <!DOCTYPE> gibt den URL für das Herunterladen der DTD für die fmresultset-Grammatik an.

Beschreibung der Elemente der fmresultset-Grammatik

Die fmresultset-Grammatik besteht hauptsächlich aus dem Element <datasource>, dem Element <metadata> und dem Element <resultset>.

Element <datasource>

In der fmresultset-Grammatik enthält das <datasource>-Element die Attribute „table“, „layout“, „date-format“, „time-format“, „timestamp-format“, „total-count“ und „database“.

- Das Attribut „date-format“ des <datasource>-Elements gibt das Format von Datumsangaben im XML-Dokument an:

  dd.MM.yyyy

Dabei gilt:

- MM ist der zweistellige Wert für den Monat (01 bis 12, wobei 01 für Januar und 12 für Dezember steht).
- dd ist der zweistellige Wert für den Tag des Monats (00 bis 31).
- yyyy ist der vierstellige Wert für das Jahr.

- Das Attribut „time-format“ des <datasource>-Elements gibt das Format von Zeitangaben im XML-Dokument an.

  HH:mm:ss

Dabei gilt:

- HH ist der zweistellige Wert für Stunden (00 bis 23 im 24-Stunden-Format).
- mm ist der zweistellige Wert für Minuten (00 bis 59).
- ss ist der zweistellige Wert für Sekunden (00 bis 59).

- Das Attribut „timestamp-format“ des <datasource>-Elements gibt das Format von Datums- und Zeitangaben in einem Zeitstempel an.

  dd.MM.yyyy HH:mm:ss

Element <metadata>

Das <metadata>-Element der fmresultset-Grammatik enthält eines oder mehrere <field-definition>- und <relatedset-definition>-Elemente, jeweils mit Attributen für eines der Felder in der Ergebnismenge. Diese Attribute geben an, ob es ein Feld mit automatischer Dateneingabe (auto-enter, „yes“ oder „no“), der maximalen Anzahl an Wiederholwerten (max-repeat), ob es ein nicht leeres Feld (not-empty, „yes“ oder „no“), ein Variablenfeld („yes“ oder „no“), ein Ergebnis („text“, „number“, „date“, „time“, „timestamp“ oder „container“), ein Typ („normal“, „calculation“ oder „summary“) ist, und sie geben den Feldnamen an (voll qualifiziert wie erforderlich).

Das <relatedset-definition>-Element repräsentiert einen Ausschnitt. Jedes Bezugsfeld in einem Ausschnitt wird durch das <field-definition>-Element innerhalb des <relatedset-definition>-Elements dargestellt. Wenn ein Ausschnitt mehrere Bezugsfelder enthält, werden die Felddefinitionen für die Bezugsfelder in einem einzigen <relatedset-definition>-Element gruppiert.

Element <resultset>

Das <resultset>-Element enthält alle <record>-Elemente, die als Ergebnis einer Query zurückgegeben wurden, sowie ein Attribut für die Gesamtzahl der gefundenen Datensätze. Jedes <record>-Element enthält die Felddaten für einen Datensatz in der Ergebnismenge – einschließlich der Attribute mod-id und record-id für den Datensatz – sowie das <data>-Element mit den Daten für ein Feld im Datensatz.

Jeder Datensatz in einem Ausschnitt wird durch ein <record>-Element innerhalb des <relatedset>-Elements dargestellt. Das Attribut „count“ des <relatedset>-Elements gibt die Anzahl der Datensätze in einem Ausschnitt an und das Attribut „table“ gibt die mit dem Ausschnitt verbundene Tabelle an.

Beispiel für XML-Daten in der fmresultset-Grammatik

Nachfolgend erhalten Sie ein Beispiel für XML-Daten, die mit der fmresultset-Grammatik generiert wurden.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE fmresultset PUBLIC "-//FMI//DTD fmresultset//EN" "/fmi/xml/fmresultset.dtd">
<fmresultset xmlns="http://www.filemaker.com/xml/fmresultset" version="1.0">
    <error code="0"></error>
    <product build="06/15/2005" name="FileMaker Web Publishing Engine" version="8.0.1.32"/>
    <datasource database="art" date-format="MM/dd/yyyy" layout="web3" table="art" time-format="HH:mm:ss"
    timestamp-format="MM/dd/yyyy HH:mm:ss" total-count="16"/>
    <metadata>
        <field-definition auto-enter="no" global="no" max-repeat="1" name="Title" not-empty="no" result="text"
        type="normal"/>
        <field-definition auto-enter="no" global="no" max-repeat="1" name="Artist" not-empty="no" result="text"
        type="normal"/>
        <relatedset-definition table="artlocation">
            <field-definition auto-enter="no" global="no" max-repeat="1" name="artlocation::Location"
            not-empty="no" result="text" type="normal"/>
            <field-definition auto-enter="no" global="no" max-repeat="1" name="artlocation::Date"
            not-empty="no" result="date" type="normal"/>
        </relatedset-definition>
        <field-definition auto-enter="no" global="no" max-repeat="1" name="Style" not-empty="no" result="text"
        type="normal"/>
        <field-definition auto-enter="no" global="no" max-repeat="1" name="length" not-empty="no"
        result="number" type="calculation"/>
    </metadata>
    <resultset count="1" fetch-size="1">
        <record mod-id="2" record-id="3">
            <field name="Title">
                <data>Still Life with Apples, Cup and Glass</data>
            </field>
            <field name="Artist">
                <data>Paul Cezanne</data>
            </field>
            <relatedset count="2" table="artlocation">
                <record mod-id="1" record-id="6">
                    <field name="artlocation::Location">
                        <data>Vault</data>
                    </field>
                    <field name="artlocation::Date">
                        <data>07.07.1997</data>
                    </field>
                </record>
                <record mod-id="0" record-id="18">
                    <field name="artlocation::Location">
                        <data>Home</data>
```

```
                    </field>
                    <field name="artlocation::Date">
                        <data>01.08.2001</data>
                    </field>
                </record>
            </relatedset>
            <field name="Style">
                <data>Impressionist</data>
            </field>
            <field name="Length">
                <data>37</data>
            </field>
        </record>
    </resultset>
</fmresultset>
```

Verwenden anderer FileMaker XML-Grammatiken

Die anderen FileMaker XML-Grammatiken enthalten Informationen über Feldtypen, Wertelisten und Layouts. FMPXMLRESULT und fmresultset sind hinsichtlich der Funktionalität äquivalent. Für den Zugriff auf Wertelisten und Feldanzeigedaten in Layouts müssen Sie die FMPXMLLAYOUT-Grammatik verwenden. Die FMPXMLRESULT- und FMPXMLLAYOUT-Grammatiken sind kompakter für den Datenaustausch.

Geben Sie zur Verwendung der FMPXMLRESULT-Grammatik den folgenden Grammatiknamen im URL an, der das XML-Dokument von der Web Publishing Engine anfordert:

FMPXMLRESULT.xml

Beispiel:

http://192.168.123.101/fmi/xml/FMPXMLRESULT.xml?–db=employees&–lay=family&–findall

Geben Sie zur Verwendung der FMPXMLLAYOUT-Grammatik den folgenden Grammatiknamen mit dem Query-Befehl -view im URL an, der das XML-Dokument von der Web Publishing Engine anfordert:

FMPXMLLAYOUT.xml

Beispiel:

http://192.168.123.101/fmi/xml/FMPXMLLAYOUT.xml?–db=employees&–lay=family&–view

Hinweis Geben Sie die FMPXMLRESULT- und die FMPXMLLAYOUT-Grammatik unbedingt in Großbuchstaben an.

Im generierten XML-Dokument verweist die Web Publishing Engine auf die Document Type Definition für die fmresultset-Grammatik in der Anweisung <!DOCTYPE> in der zweiten Zeile des Dokuments, unmittelbar nach der Anweisung <?xml...?>. Die Anweisung <!DOCTYPE> gibt den URL für das Herunterladen der DTD für die Grammatik an.

Beschreibung der Elemente in der FMPXMLRESULT Grammatik

In der FMPXMLRESULT-Grammatik enthält das <DATABASE>-Element die Attribute NAME, RECORDS, DATEFORMAT, LAYOUT und TIMEFORMAT.

Das Attribut DATEFORMAT des <DATABASE>-Elements gibt das Datumsformat im XML-Dokument an. Das Attribut TIMEFORMAT des <DATABASE>-Elements gibt das Zeitformat im XML-Dokument an. Datums- und Zeitformate der Grammatiken FMPXMLRESULT und fmresultset sind gleich (siehe dazu die Tabellen unter „Beschreibung der Elemente der fmresultset-Grammatik“ auf Seite 33.)

Das <METADATA>-Element der FMPXMLRESULT-Grammatik enthält eines oder mehrere <FIELD>-Elemente. Jedes FIELD-Element enthält Daten für ein Feld bzw. eine Spalte in der Ergebnismenge – einschließlich des Feldnamens, wie er in der Datenbank definiert ist –, den Feldtyp, die Ja/Nein-Zulässigkeit für leere Felder (Attribut EMPTYOK) und die maximale Anzahl von Wiederholwerten (Attribut MAXREPEAT). Gültige Werte für Feldtypen sind TEXT, NUMBER, DATE, TIME, TIMESTAMP und CONTAINER.

Das <RESULTSET>-Element enthält alle <ROW>-Elemente, die als Ergebnis einer Query zurückgegeben wurden, sowie ein Attribut für die Gesamtzahl der gefundenen Datensätze. Jedes <ROW>-Element enthält die Feld-/Spalten-Daten für jede Zeile in der Ergebnismenge. Diese Daten umfassen RECORDID und MODID für die Zeile (siehe „Query-Parameter –modid (Änderungs-ID)“ auf Seite 107) und das <COL>-Element. Das <COL>-Element enthält die Daten für ein Feld/eine Spalte in der Zeile, wobei mehrere <DATA>-Elemente einen der Werte in einem Wiederhol- oder Ausschnittfeld repräsentieren.

Beispiel für XML-Daten in der FMPXMLRESULT Grammatik

Nachfolgend sehen Sie ein Beispiel für XML-Daten, die mit der FMPXMLRESULT-Grammatik generiert wurden.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE FMPXMLRESULT PUBLIC "-//FMI//DTD FMPXMLRESULT//EN" "/fmi/xml/FMPXMLRESULT.dtd">
<FMPXMLRESULT xmlns="http://www.filemaker.com/fmpxmlresult">
    <ERRORCODE>0</ERRORCODE>
    <PRODUCT BUILD="06/15/2005" NAME="FileMaker Web Publishing Engine" VERSION="8.0.1.32"/>
    <DATABASE DATEFORMAT="MM/dd/yyyy" LAYOUT="web3" NAME="art" RECORDS="16"
    TIMEFORMAT="HH:mm:ss"/>
    <METADATA>
        <FIELD EMPTYOK="YES" MAXREPEAT="1" NAME="Title" TYPE="TEXT" />
        <FIELD EMPTYOK="YES" MAXREPEAT="1" NAME="Artist" TYPE="TEXT" />
        <FIELD EMPTYOK="YES" MAXREPEAT="1" NAME="artlocation::Location" TYPE="TEXT"/>
        <FIELD EMPTYOK="YES" MAXREPEAT="1" NAME="artlocation::Date" TYPE="DATE"/>
        <FIELD EMPTYOK="YES" MAXREPEAT="1" NAME="Style" TYPE="TEXT"/>
        <FIELD EMPTYOK="YES" MAXREPEAT="1" NAME="Length" TYPE="NUMBER"/>
    </METADATA>
    <RESULTSET FOUND="1">
        <ROW RECORDID="3" MODID="23">
            <COL>
                <DATA>The Dancers in Blue</DATA>
            </COL>
            <COL>
                <DATA>Edgar Degas</DATA>
```

```
            </COL>
            <COL>
                <DATA>Study</DATA>
            </COL>
            <COL>
                <DATA>01/08/1979</DATA>
            </COL>
            <COL>
                <DATA>Impressionist</DATA>
            </COL>
            <COL>
                <DATA>19</DATA>
            </COL>
        </ROW>
    </RESULTSET>
</FMPXMLRESULT>
```

Die Reihenfolge der <COL>-Elemente entspricht der Reihenfolge der <FIELD>-Elemente im <METADATA>-Element – z.B. entspricht die Reihenfolge der Felder „Title“ und „Artist“ im <METADATA>-Element der Reihenfolge von „The Dancers in Blue“ und „Edgar Degas“ in den Elementen <RESULTSET> und <ROW>.

Beschreibung der Elemente der Grammatik FMPXMLLAYOUT

In der FMPXMLLAYOUT-Grammatik enthält das <LAYOUT>-Element den Namen des Layouts, den Namen der Datenbank sowie <FIELD>-Elemente für jedes Feld im entsprechenden Layout der Datenbank. Jedes <FIELD>-Element beschreibt den Feldtyp und enthält das Attribut VALUELIST für jede Werteliste des Feldes.

Das <VALUELISTS>-Element enthält eines oder mehrere <VALUELIST>-Elemente für jede Werteliste im Layout – jedes VALUELIST-Element enthält den Namen der Werteliste und ein <VALUE>-Element für jeden Wert in der Liste.

Für Datums-, Zeit- und Zeitstempelfelder werden Daten für Wertelisten über das „fm“-Format für diesen Feldtyp formatiert. Die „fm“-Formate sind MM/tt/jjjj für Datum, HH:mm:ss für Zeit und MM/tt/jjjj HH:mm:ss für Zeitstempel. Weitere Informationen finden Sie unter „Verwenden der Datums-, Zeit- und Tages-Erweiterungsfunktionen“ auf Seite 81. Wenn zum Beispiel eine Werteliste „Geburtstage“ für ein Einblendmenü für ein Feld „Geburtstag“ eines Layouts verwendet wird und das Feld „Geburtstag“ ein Datumsfeld ist, werden alle für diese Werteliste ausgegebenen Werte im Datumsformat „fm“ ausgegeben.

Hinweis Wenn zwei Felder mit unterschiedlichem Feldtyp in einem Layout die gleiche Werteliste verwenden, bestimmt das erste Feld das Format der Daten der Werteliste.

XML-Daten in der FMPDSORESULT-Grammatik – Beispiel

Im Folgenden finden Sie ein Beispiel für XML-Daten, die mit der FMPXMLLAYOUT-Grammatik generiert wurden.

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE FMPXMLLAYOUT PUBLIC "-//FMI//DTD FMPXMLLAYOUT//EN" "/fmi/xml/FMPXMLLAYOUT.dtd">
<FMPXMLLAYOUT xmlns="http://www.filemaker.com/fmpxmllayout">
    <ERRORCODE>0</ERRORCODE>
    <PRODUCT BUILD="06/15/2005" NAME="FileMaker Web Publishing Engine" VERSION="8.0.1.32"/>
    <LAYOUT DATABASE="art" NAME="web2">
        <FIELD NAME="Title">
            <STYLE TYPE="EDITTEXT" VALUELIST="" />
        </FIELD>
        <FIELD NAME="Artist">
            <STYLE TYPE="EDITTEXT" VALUELIST="" />
        </FIELD>
        <FIELD NAME="Image">
            <STYLE TYPE="EDITTEXT" VALUELIST="" />
        </FIELD>
        <FIELD NAME="artlocation::Location">
            <STYLE TYPE="EDITTEXT" VALUELIST="" />
        </FIELD>
        <FIELD NAME="artlocation::Date">
            <STYLE TYPE="EDITTEXT" VALUELIST="" />
        </FIELD>
        <FIELD NAME="Style">
            <STYLE TYPE="POPUPMENU" VALUELIST="style"/>
        </FIELD>
    </LAYOUT>
    <VALUELISTS>
        <VALUELIST NAME="style">
            <VALUE>Impressionist</VALUE>
            <VALUE>Modern</VALUE>
            <VALUE>Abstract</VALUE>
        </VALUELIST>
    </VALUELISTS>
</FMPXMLLAYOUT>
```

Erläuterung von UTF-8-kodierten Daten

Alle von der Web Publishing Engine generierten XML-Daten werden im UTF-8-Format (Unicode Transformation Format 8) kodiert. Dieses Format komprimiert ASCII-Zeichen im Unicode-Standardformat von 16 Bit auf 8 Bit. Für Unicode und UFT-8 sind XML-Parser erforderlich.

UTF-8-Kodierung unterstützt die direkte Darstellung der Werte 0-127 für den ASCII-Standardzeichensatz in der englischen Sprache und bietet Multibyte-Kodierungen für Unicode-Zeichen mit höheren Werten.

Hinweis Stellen Sie sicher, dass Sie einen Webbrowser oder ein Textverarbeitungsprogramm verwenden, das UTF-8-Dateien unterstützt.

Das Kodierungsformat UTF-8 hat die folgenden Eigenschaften:

- Alle ASCII-Zeichen sind UTF-8-Zeichen von der Größe eines Byte. Eine gültige ASCII-Zeichenkette ist eine gültige UTF-8-Zeichenkette.
- Jedes Nicht-ASCII-Zeichen (beliebiges Zeichen mit gesetztem höherwertigem Bit) ist Teil eines Multibyte-Zeichens.
- Das erste Byte eines UTF-8-Zeichens gibt die Anzahl der zusätzlichen Bytes im Zeichen an.
- Das erste Byte eines Multibyte-Zeichens lässt sich leicht vom darauf folgenden Byte unterscheiden, wodurch sich der Beginn eines Zeichens von einer beliebigen Position in einem Datenstrom einfach ermitteln lässt.
- Eine Konvertierung zwischen UTF-8 und Unicode ist einfach vorzunehmen.
- Die UTF-8-Kodierung ist relativ kompakt. Bei Text, der einen hohen Prozentsatz an ASCII-Zeichen enthält, ist sie kompakter als Unicode. Im schlimmsten Fall ist ein UTF-8-String nur 50 % größer als der entsprechende Unicode-String.

Abfragen von XML-Daten mithilfe von FileMaker-Query-Strings

Um XML-Daten aus einer FileMaker-Datenbank abzufragen, verwenden Sie die FileMaker-Query-Befehle und -Parameter in einem Query-String. Sie können beispielsweise den Query-Befehl –findall im folgenden Query-String in einem URL verwenden, um eine Liste aller Produkte aus einer FileMaker-Datenbank mit dem Namen products anzufordern:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=products–lay=sales&–findall

Ein Query-String darf nur einen Query-Befehl, z. B. –new, enthalten. Die meisten Query-Befehle verlangen auch mehrere entsprechende Query-Parameter im Query-String. Beispielsweise verlangen alle Query-Befehle außer –dbnames den Parameter –db, der die abzufragende Datenbank angibt.

Sie können auch Query-Befehle und -Parameter in einem URL oder in einer <?xslt–cwp–query?>-Verarbeitungsanweisung in einem FileMaker XSLT-Stylesheet verwenden. Weitere Informationen finden Sie unter Kapitel 5, „Entwickeln von FileMaker XSLT-Stylesheets“.

Dieser Abschnitt bietet eine Zusammenfassung der FileMaker Query-Befehle und -Parameter. Weitere Informationen über ihre Verwendung in einem Query-String finden Sie in Anhang A, „Gültige Namen in Query-Strings“.

Hinweis Die Web Publishing Engine unterstützt auch einen zusätzlichen Query-Befehl (–process) sowie drei Query-Parameter, die nur zur Verwendung mit FileMaker XSLT-Stylesheets definiert sind. Weitere Informationen finden Sie unter „Verwenden von Query-Strings in FileMaker XSLT-Stylesheets“ auf Seite 60.

Verwenden Sie diesen Query-Befehlsnamen	Zur Ausführung dieses Befehls
–dbnames	Namen aller bereitgestellten und im Web gemeinsam genutzten Datenbanken abrufen
–delete	Datensatz löschen
–dup	Datensatz duplizieren
–edit	Datensatz bearbeiten
–find	Datensatz/Datensätze suchen
–findall	Nach allen Datensätzen suchen
–findany	Zufällig ermittelten Datensatz anzeigen
–layoutnames	Namen aller verfügbaren Layouts für eine bereitgestellte, im Web gemeinsam genutzte Datenbank abrufen
–new	Neuer Datensatz
–scriptnames	Namen aller verfügbaren Scripts für eine bereitgestellte, im Web gemeinsam genutzte Datenbank abrufen
–view	Layout-Informationen aus einer Datenbank abrufen, wenn die FMPXMLLAYOUT-Grammatik angegeben ist. Ruft den Abschnitt <metadata> des XML-Dokuments und eine leere Datensatzmenge ab, wenn die fmresultset- oder FMPXMLRESULT-Grammatik angegeben ist.

Verwenden Sie diese Query-Parameternamen	Mit diesen Query-Befehlen
–db (Datenbankname)	Erforderlich für alle Query-Befehle mit Ausnahme von –dbnames und –process (nur XSLT-Abfragen)
–field	Erforderlich zur Angabe eines Felds in einem URL für Medienabfragen. Weitere Informationen finden Sie unter „Erläuterung der URL-Syntax für FileMaker-Medienobjekte in XML-Lösungen“ auf Seite 30.
Fieldname	Mindestens ein Feldname ist für –edit erforderlich. Optional bei –find. Weitere Informationen finden Sie unter „Query-Parameter „fieldname“ (Nicht-Medienfeldname)“ auf Seite 104.
Fieldname.op (Operator)	Optional bei –find
–lay (Layoutname)	Erforderlich für alle Query-Befehle mit Ausnahme von –dbnames, –layoutnames, –scriptnames und –process (nur XSLT-Abfragen)
–lay.response (Layout für XML-Antwort wechseln)	Optional für alle Query-Befehle mit Ausnahme von –dbnames, –layoutnames, –scriptnames und –process (nur XSLT-Abfragen)
–lop (logischer Operator)	Optional bei –find

Verwenden Sie diese Query-Parameternamen	Mit diesen Query-Befehlen
–max (maximale Datensätze)	Optional bei –find, –findall
–modid (Änderungs-ID)	Optional bei –edit
–recid (Datensatz-ID)	Optional bei –edit, –delete, –dup. Optional bei –find
–script (Script ausführen)	Optional bei –find, –findall, –findany, –new, –edit, –delete, –dup, –view
–script.param (einen Parameterwert an das durch –script angegebene Script übergeben)	Optional bei –script
–script.prefind (Script vor –find, –findany und –findall ausführen)	Optional bei –find, –findany, –findall
–script.prefind.param (einen Parameterwert an das durch –script.prefind angegebene Script übergeben)	Optional bei –script.prefind
–script.presort (Script vor Sortierung ausführen)	Optional bei –find, –findall
–script.presort.param (einen Parameterwert an das durch –script.presort angegebene Script übergeben)	Optional bei –script.presort
–skip (Datensätze überspringen)	Optional bei –find, –findall
–sortfield.[1-9] (Sortierfeld)	Optional bei –find, –findall
–sortorder.[1-9] (Sortierfolge)	Optional bei –find, –findall
–stylehref (Stylesheet HREF)	Optional für alle Query-Befehle (zur Angabe eines Stylesheet-URLs für –styletype)
–styletype (Stylesheet-Typ)	Optional für alle Query-Befehle (zur Angabe eines Client-seitigen Stylesheets)

Wechseln des Layouts für eine XML-Antwort

Der Query-Parameter –lay gibt das Layout an, das Sie beim Abfragen von XML-Daten verwenden wollen. Häufig eignet sich dasselbe Layout für die Verarbeitung der Daten, die von der Abfrage geliefert werden. In einigen Fällen sollten Sie die Daten in einem Layout durchsuchen, das Felder enthält, die aus Sicherheitsgründen in keinem Layout vorhanden sind, mit dem die Ergebnisse angezeigt werden. (Um Daten in einem Feld zu suchen, muss das Feld auf das Layout gesetzt werden, das Sie in der XML-Abfrage angeben.)

Um für die Anzeige der XML-Antwort ein anderes Layout als für die Verarbeitung der XML-Abfrage zu verwenden, können Sie den optionalen Query-Parameter –lay.response angeben.

Die folgende Abfrage sucht z. B. Werte größer 100.000 im Feld Salary im Layout Budget. Die gefundenen Daten werden im Layout ExecList angezeigt, das das Feld Salary nicht enthält.

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=Budget&Salary=100000&Salary.op=gt&–find&–lay.response=ExecList

Erläuterung der Verarbeitung einer XML-Abfrage

Es gibt mehrere Query-Parameter, die die Verarbeitung einer XML-Abfrage und die Generierung eines XML-Dokuments beeinflussen.

FileMaker Server und die Web Publishing Engine verarbeiten eine XML-Abfrage in dieser Reihenfolge:

1. Verarbeitung des –lay -Query-Parameters
2. Festlegen der globalen Feldwerte, die in der Query angegeben sind (der Teil .global= des URLs)
3. Verarbeitung des Query-Parameters –script.prefind, falls angegeben
4. Verarbeitung der Query-Befehle wie z. B. –find oder –new
5. Verarbeitung des Query-Parameters –script.presort, falls angegeben
6. Sortieren der Ergebnisdaten, falls eine Sortierung angegeben wurde
7. Verarbeitung des Query-Parameters –lay.response, um das Layout zu wechseln, falls angegeben
8. Verarbeitung des Query-Parameters –script, falls angegeben
9. Generieren des XML-Dokuments.

Wenn einer der obigen Schritte einen Fehlercode erzeugt, stoppt die Abfrageverarbeitung und etwaige nachfolgende Schritte werden nicht ausgeführt. Jedoch bleiben alle vorangehenden Schritte in der Abfrage ausgeführt.

Betrachten Sie z. B. eine Abfrage, die den aktuellen Datensatz löscht, die Datensätze sortiert und dann ein Script ausführt. Wenn der Parameter –sortfield ein nicht vorhandenes Feld angibt, löscht die Abfrage den aktuellen Datensatz und gibt dann Fehlercode 102 (Feld fehlt) zurück, führt aber das Script nicht aus.

Verwenden von Server-seitiger und Client-seitiger Stylesheet-Verarbeitung

Die Web Publishing Engine unterstützt Server-seitige Verarbeitung eines XSLT-Stylesheets und gestattet Ihnen auch die Verwendung eines Query-Parameters, der Client-seitige Stylesheet-Verarbeitung angibt.

Es ist wichtig, die Unterschiede zwischen den beiden Arten der Stylesheet-Verarbeitung sowie die Implikationen für die Sicherheit bei Client-seitiger Verarbeitung zu verstehen. Server-seitige Verarbeitung gibt Web-Benutzern keinen Zugang zu den ungefilterten XML-Daten und ist daher sicherer als Client-seitige Verarbeitung. Bei Server-seitiger Verarbeitung werden die Daten in einer Form präsentiert, die der Eigentümer der Daten oder der Verfasser des XSLT-Stylesheets für angemessen hält. Server-seitige Verarbeitung verbirgt die Datenbanknamen, Feldnamen und andere Implementierungsdetails vor Web-Benutzern. Server-seitige Verarbeitung kann auch für die Angabe statisch definierter Query-Parameter benutzt werden, die die Verwendung nicht zugelassener Query-Befehle und Query-Parameter verhindert, z. B. Datenbanknamen. Weitere Informationen hierzu finden Sie unter Kapitel 4, „Einführung in Custom Web Publishing mit XSLT“ und Kapitel 5, „Entwickeln von FileMaker XSLT-Stylesheets“.

Wenn Ihre Lösung Client-seitige Stylesheet-Verarbeitung erfordert, können Sie über die Web Publishing Engine eine Anweisung für die XML-Stylesheet-Verarbeitung mit jeder Grammatik generieren, indem Sie die Parameter –styletype und –stylehref in die FileMaker-Query-Stringabfrage einfügen. Sie können Cascading Style Sheets (CSS) oder XSLT-Stylesheets für die Anzeige Ihres XML-Dokuments verwenden.

- Mit dem Parameter –styletype wird der Wert des type-Attributs festgelegt (type=text/css oder type=text/xsl).
- Mit dem Parameter –stylehref wird der Wert des HREF-Attributs festgelegt, das die Position des Stylesheets mit einem absoluten Pfad angibt. Beispiel: href=/mystylesheet.css oder href=/stylesheets/mystylesheet.xsl Das Stylesheet kann einen beliebigen Namen haben, aber die Erweiterung muss .css oder .xsl sein.

Beispiel für einen FileMaker-Query-String, der Client-seitige Stylesheet-Verarbeitung erzeugt:

```
http://localhost/fmi/xml/fmresultset.xml?–db=products–lay=sales&–findall&–styletype=text/xsl
&–stylehref=/mystylesheet.xsl
```

Hinweis Der Schrägstrich / in –stylehref=/document.xsl in diesem Beispiel wird verwendet, da das Stylesheet sich im Root-Ordner der Web-Server-Software befindet. Verwenden Sie für das Stylesheet einen URL mit einem absoluten Pfad, um die Position auf dem Web-Server anzugeben. Das Stylesheet kann sich auch auf einem anderen Web-Server befinden.

Auf der Grundlage dieser Abfrage nimmt die Web Publishing Engine die folgende Verarbeitungsanweisung in das XML-Dokument auf:

```
<?xml–stylesheet type="text/xsl" href="/mystylesheet.xsl"?>
```

Kopieren oder platzieren Sie das Stylesheet für Client-seitige Verarbeitung auf den Web-Server an die Stelle, die der absolute Pfad im URL für das HREF-Attribut angibt.

Wichtig Platzieren Sie Stylesheets für Client-seitige Verarbeitung nicht in den Ordner xslt-template-files, der für Server-seitige XSLT-Stylesheet-Verarbeitung verwendet wird. Weitere Informationen finden Sie unter „Verwenden von FileMaker XSLT-Stylesheets in einer Website oder einem Programm“ auf Seite 54.

Hinweis Einige Webbrowser unterstützen keine Client-seitige Verarbeitung. Weitere Informationen finden Sie in der Dokumentation zu Ihrem Webbrowser.

Fehlerbehebung beim XML-Dokumentzugriff

Wenn beim Zugriff auf XML-Dokumente mit der Web Publishing Engine Probleme auftreten, prüfen Sie Folgendes:

- Die erweiterten Zugriffsrechte in der Datenbank müssen auf „XML Custom Web Publishing“ eingestellt und einem Benutzerkonto zugewiesen sein. Weitere Informationen finden Sie unter „Aktivieren von Instant Web Publishing in einer Datenbank“ auf Seite 21.
- Die Datenbank muss über FileMaker Server bereitgestellt und geöffnet sein. Weitere Informationen finden Sie in der FileMaker Server Administration Hilfe
- Prüfen Sie, ob der verwendete Datenbankkontoname und ggf. Ihr Passwort korrekt sind.
- Stellen Sie sicher, dass Web-Server und Web Publishing Engine laufen.
- XML-Publishing muss in der Web Publishing Engine aktiviert sein. Weitere Informationen finden Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.

Kapitel 4
Einführung in Custom Web Publishing mit XSLT

FileMaker XSLT-Stylesheets ermöglichen das Umwandeln, Filtern oder Formatieren von XML-Daten für die Verwendung in einem Webbrowser oder in anderen Programmen und Anwendungen. Dieses Kapitel stellt FileMaker XSLT-Stylesheets und zwei Tools vor, die Ihnen den Einstieg in die Erstellung von XSLT-Stylesheets erleichtern: den Site Assistant und den CDML Converter. Weitere Informationen über den Aufbau von FileMaker XSLT-Stylesheets finden Sie in Kapitel 5, „Entwickeln von FileMaker XSLT-Stylesheets“.

Erläuterung von FileMaker XSLT-Stylesheets

Mithilfe von FileMaker XSLT-Stylesheets können Sie:

- FileMaker-Daten filtern, indem Sie steuern, welche Datenbankfelder vom Stylesheet veröffentlicht werden.
- Metadaten wie z. B. Datenbank- und Feldnamen ausblenden.
- das Erscheinungsbild der Daten in einer Web-Seite formatieren und steuern, wie der Web-Benutzer mit den Daten interagiert.
- die Daten als HTML oder Text ausgeben, z. B. vCards oder durch Komma getrennte Werte.
- die Daten von einer FileMaker XML-Grammatik in eine andere XML-Grammatik umwandeln, um sie in einer anderen Datenbank oder Anwendung zu nutzen, z. B. Scalable Vector Graphics (SVG).
- eine Untergruppe der FileMaker-Daten in andere Websites und mit anderen Middleware- und Spezialanwendungen integrieren, die sich erheblich von der FileMaker-Datenbank unterscheiden können.
- die veröffentlichten Feldnamen ändern und unbefugte Verwendung der Datenbankdesigninformation verhindern.

Hinweis Custom Web Publishing mit XSLT für FileMaker Server basiert auf der W3C-Empfehlung für XSLT 1.0. Informationen über XSLT 1.0 finden Sie unter www.w3.org. Zusätzliche Funktionalität wie Sitzungsverwaltung, E-Mail-Versand und Zugriff auf Cookies und Header wird durch FileMaker XSLT-Erweiterungsfunktionen zur Verfügung gestellt. Weitere Informationen finden Sie unter „Verwenden der FileMaker XSLT-Erweiterungsfunktionen und -parameter“ auf Seite 65. Die Web Publishing Engine unterstützt nicht XSL Formatting Objects (XSL-FO).

Beispiele für die Verwendung von FileMaker XSLT-Stylesheets

Ein paar Beispiele für die unzähligen Möglichkeiten mit FileMaker XSLT-Stylesheets:

- Sie können eine Tabelle in eine Web-Seite mit einem Teil der Daten aus einer FileMaker-Datenbank einfügen, damit Web-Benutzer darin blättern können. Die Tabelle könnte z. B. Namen und Adressen, aber keine Telefonnummern enthalten. Um unbefugten Zugriff zu verhindern, kann die Web-Seite statt der tatsächlichen Feldnamen in der FileMaker-Datenbank (z. B. „vor_name“) generische Beschriftungen für die Daten zeigen (z. B. „Name“).
- Sie können eine Web-Seite oder eine Anwendung erstellen, die Daten aus einem FileMaker-Ausschnitt in Informationen aus anderen Datenquellen integriert.

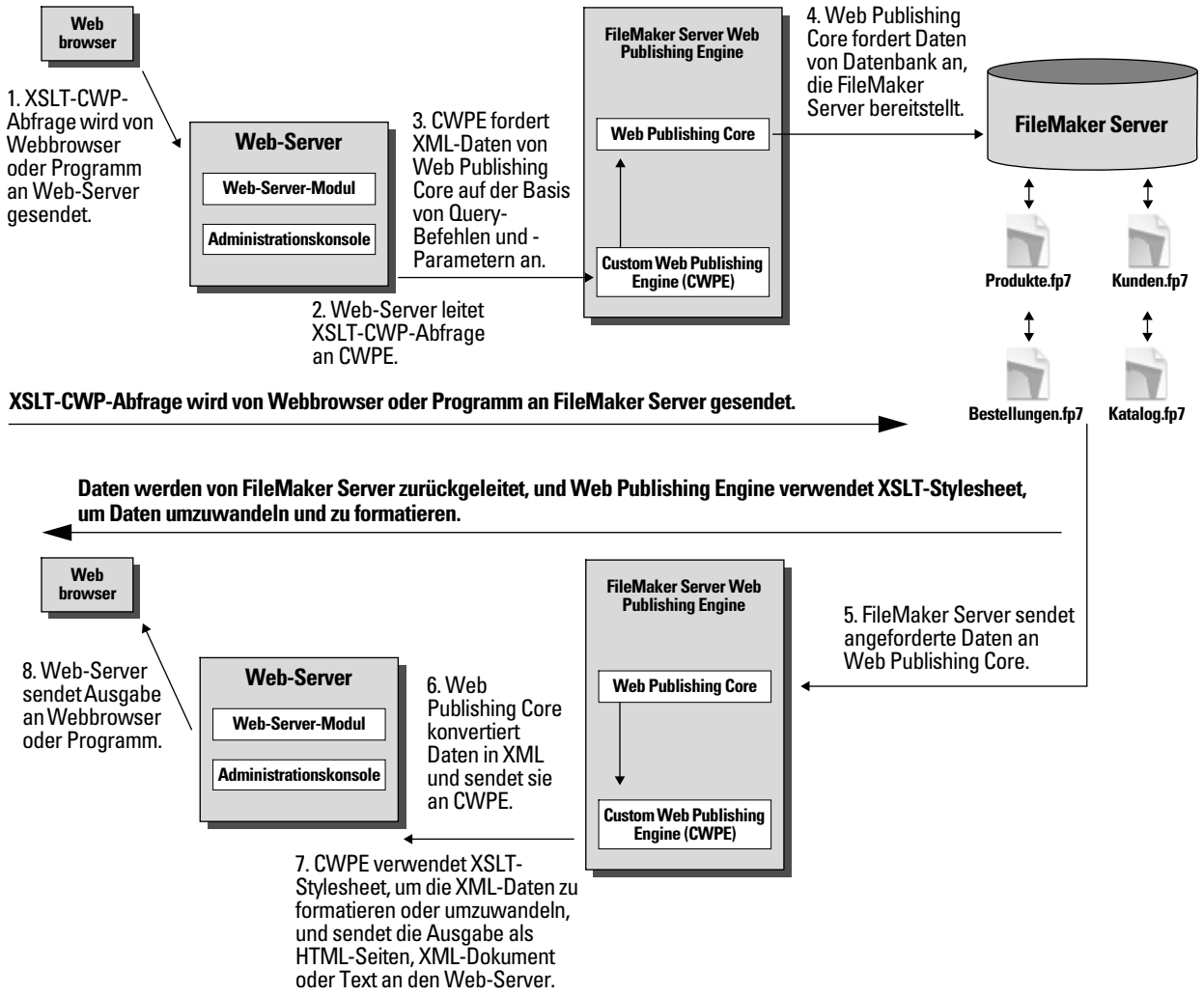
- Sie können einer Web-Seite eine Taste hinzufügen, über die eine vCard aus den Kontaktinformationen einer Person in einer FileMaker-Datenbank erzeugt wird.
- Sie können die XML-Daten aus einer FileMaker-Datenbank in eine XML-Grammatik umwandeln, die eine Tabellenkalkulations- oder Datenbankanwendung öffnen kann.

Einstieg in die Verwendung von Custom Web Publishing mit XSLT

Wenn Sie Standard-XML und XSLT kennen, können Sie sofort mit dem Einsatz der Web Publishing Engine beginnen, nachdem Sie ein paar spezielle Details über FileMaker XML und XSLT wie die Verwendung von FileMaker-spezifischen XSLT-Erweiterungsfunktionen, Query-Befehlen und Query-Parametern gelernt haben. Site Assistant und CDML Converter sind Tools, die Ihnen die ersten Schritte beim Erstellen von Stylesheets erleichtern und Ihnen zeigen, wie diese aufgebaut werden. Sie können die Stylesheets mithilfe Ihres bevorzugten XML- und XSLT-Erstellungswerkzeugs weiter ausarbeiten.

Wie die Web Publishing Engine auf der Basis von XML-Daten und XSLT-Stylesheets Seiten generiert

Nachdem eine XSLT Custom Web Publishing (XSLT-CWP)-Abfrage an den Web-Server gesendet wurde, fragt die Web Publishing Engine die FileMaker-Datenbank auf der Basis der Query-Befehle und -Parameter ab, die im Stylesheet und im URL definiert sind, und gibt die Daten dann gemäß den Anweisungen im XSLT-Stylesheet aus.


Web browser
1. XSLT-CWP-Abfrage wird von Webbrowser oder Programm an Web-Server gesendet.
Web-Server
Web-Server-Modul
Administrationskonsole
2. Web-Server leitet XSLT-CWP-Abfrage an CWPE.
3. CWPE fordert XML-Daten von Web Publishing Core auf der Basis von Query-Befehlen und -Parametern an.
FileMaker Server Web Publishing Engine
Web Publishing Core
Custom Web Publishing Engine (CWPE)
4. Web Publishing Core fordert Daten von Datenbank an, die FileMaker Server bereitstellt.
FileMaker Server
Produkte.fp7
Kunden.fp7
Bestellungen.fp7
Katalog.fp7
XSLT-CWP-Abfrage wird von Webbrowser oder Programm an FileMaker Server gesendet.
Daten werden von FileMaker Server zurückgeleitet, und Web Publishing Engine verwendet XSLT-Stylesheet, um Daten umzuwandeln und zu formatieren.
Web browser
8. Web-Server sendet Ausgabe an Webbrowser oder Programm.
Web-Server
Web-Server-Modul
Administrationskonsole
6. Web Publishing Core konvertiert Daten in XML und sendet sie an CWPE.
7. CWPE verwendet XSLT-Stylesheet, um die XML-Daten zu formatieren oder umzuwandeln, und sendet die Ausgabe als HTML-Seiten, XML-Dokument oder Text an den Web-Server.
FileMaker Server Web Publishing Engine
Web Publishing Core
Custom Web Publishing Engine (CWPE)
5. FileMaker Server sendet angeforderte Daten an Web Publishing Core.

Allgemeine Schritte bei der Verwendung von Custom Web Publishing mit XSLT

Zusammenfassung der Schritte bei der Verwendung von Custom Web Publishing mit XSLT:

1. Stellen Sie in der Web Publishing Engine-Administrationskonsole sicher, dass „XSLT Publishing“ aktiviert ist Weitere Informationen erhalten Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.
2. Öffnen Sie in FileMaker Pro jede FileMaker-Datenbank, die Sie veröffentlichen wollen, und stellen Sie sicher, dass in der Datenbank das erweiterte Zugriffsrecht fmxslt für Custom Web Publishing mit XSLT aktiviert ist. Weitere Informationen finden Sie unter „Aktivieren von Instant Web Publishing in einer Datenbank“ auf Seite 21.

   Hinweis Stellen Sie sicher, dass Sie entsprechende FileMaker-Datenbankberechtigungen verwenden, wenn Sie Stylesheets entwickeln, die an den Endbenutzer gegeben werden. Andernfalls haben Sie in der FileMaker-Datenbank eventuell Zugriff auf Layouts und Funktionen, die dann dem Endbenutzer nicht zur Verfügung stehen und damit zu inkonsistentem Verhalten führen.
3. Erstellen Sie XSLT-Stylesheets, die FileMaker-spezifische XSLT-Erweiterungsfunktionen, Query-Befehle und Query-Parameter enthalten, um die XML-Daten aus einer FileMaker-Datenbank zu formatieren oder umzuwandeln.

   Sie können mithilfe des FileMaker-Tools Site Assistant eine oder mehrere grundlegende XSLT-Stylesheets als Ausgangsposition für Ihre Site erstellen. Weitere Informationen finden Sie im nächsten Abschnitt, „Generieren von XSLT-Stylesheets mithilfe des FileMaker Site Assistant“.

   Falls CDML-Lösungen vorhanden sind, können Sie die Dateien im CDML-Format mithilfe von CDML Converter in XSLT-Stylesheets umwandeln. Weitere Informationen finden Sie unter „Verwenden von FileMaker CDML Converter“ auf Seite 50.

   Sie können auch Ihr eigenes XSLT-Erstellungs- oder Textbearbeitungswerkzeug verwenden, um die XSLT-Stylesheets wie erforderlich zu ändern oder Ihre Stylesheets ganz neu zu entwickeln. Weitere Informationen finden Sie unter Kapitel 5, „Entwickeln von FileMaker XSLT-Stylesheets“.
4. Kopieren oder platzieren Sie die XSLT-Stylesheets im Ordner xslt-template-files, der sich im Ordner Web Publishing im Ordner FileMaker Server auf dem Host befindet, auf dem die Web Publishing Engine installiert ist.

   Sie können die Stylesheets auch in einem optionalen Ordner oder einer Ordnerhierarchie innerhalb des Ordners xslt-template-files platzieren.
5. Platzieren Sie etwaige statische Dateien auf dem Web-Server. Weitere Informationen finden Sie unter „Verwenden von FileMaker XSLT-Stylesheets in einer Website oder einem Programm“ auf Seite 54.
6. Erstellen oder ändern Sie eine Website oder ein Programm, das die XSL-Stylesheets verwendet.

   Sie können z. B. für die Website eine statische Seite wie index.html verwenden, die entweder Web-Benutzer automatisch an ein XSLT-Stylesheet weiterleitet oder einen Link auf das XSLT-Stylesheet enthält.
7. Richten Sie unbedingt Sicherheitsmechanismen für Ihre Site oder Ihr Programm ein. Weitere Informationen finden Sie im *FileMaker Sicherheitshandbuch*.
8. Testen Sie die Site bzw. das Programm mit den XSLT-Stylesheets unter Verwendung der Konten und Berechtigungen, die für Web-Benutzer definiert sind.
9. Geben Sie die Site bzw. das Programm frei und informieren Sie Benutzer.

Generieren von XSLT-Stylesheets mithilfe des FileMaker Site Assistant

Der FileMaker Site Assistant ist eine Anwendung, mit der Sie grundlegende XSLT-Stylesheets als Startpunkt für die Verwendung mit Custom Web Publishing mit XSLT erstellen können. Der Site Assistant eignet sich ideal, um den Aufbau von FileMaker XSLT-Stylesheets kennen zu lernen. Sie können dann Ihr eigenes Erstellungswerkzeug für XSLT-Stylesheets oder Textverarbeitungswerkzeuge verwenden, um die Stylesheets je nach Bedarf zu verändern. Sie können den Site Assistant nicht zur Bearbeitung oder Aktualisierung bestehender Stylesheets verwenden, aber Sie können ihn verwenden, um die anfänglichen Stylesheets für eine gesamte Site zu generieren oder ein einzelnes Stylesheet zu erzeugen und damit einer bestehenden Site zusätzliche Funktionalität (z. B. das Löschen von Datensätzen) zu geben.

Mithilfe des Site Assistant können Sie XSLT-Stylesheets für alle Arten von Seiten generieren, die nützlich für die Arbeit mit FileMaker-Datenbanken über Custom Web Publishing sind. Abhängig von dem im Site Assistant gewählten Optionen können Sie eine Site erstellen, die Benutzern das Folgende erlaubt:

- Anzeige einzelner Datensätze
- Anzeige einer Liste mit allen Datensätzen in der Datenbank
- Durchsuchen der Datenbank und Anzeigen des Ergebnisses in einer Liste
- Sortieren von Datensätzen
- Hinzufügen von Datensätzen
- Bearbeiten und Duplizieren von Datensätzen
- Löschen von Datensätzen
- Anzeigen eines Statistikberichts

Ferner können Sie eine optionale Homepage generieren, die mit den anderen generierten XSLT-Stylesheet-Seiten verknüpft ist.

Die Web Publishing Engine verwendet jedes Ihrer Stylesheets, um Daten dynamisch aus einer FileMaker-Datenbank zu beziehen, wenn ein Web-Benutzer eine HTTP-Anforderung und einen URL sendet, der eines Ihrer XSLT-Stylesheets referenziert. Die Web Publishing Engine verwendet ein Stylesheet, um die XML-Daten umzuwandeln und zu formatieren, und generiert die resultierende HTML-Seite, mit der der Web-Benutzer arbeiten kann.

Hinweis Die Site Assistant-Stylesheets transformieren FileMaker XML-Daten basierend auf der XML-Grammatik fmresultset in HTML-Seiten, so dass die Stylesheets nicht kompatibel mit anderer Verwendung von XML-Daten wie dem FileMaker XML-Export sind.

Installieren des Site Assistant

Informationen über die Installation des Site Assistant finden Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.

Vor Verwendung des Site Assistant

Bevor Sie den Site Assistant verwenden können, um XSLT-Stylesheets für eine Datenbank zu generieren:

- Setzen Sie in der Datenbank das erweiterte Zugriffsrecht fmxslt. Verwenden Sie bei der Ausführung des Web Assistant Berechtigungen, die denen entsprechen, die Sie den Web-Benutzern geben. Weitere Informationen finden Sie unter „Aktivieren von Instant Web Publishing in einer Datenbank“ auf Seite 21.

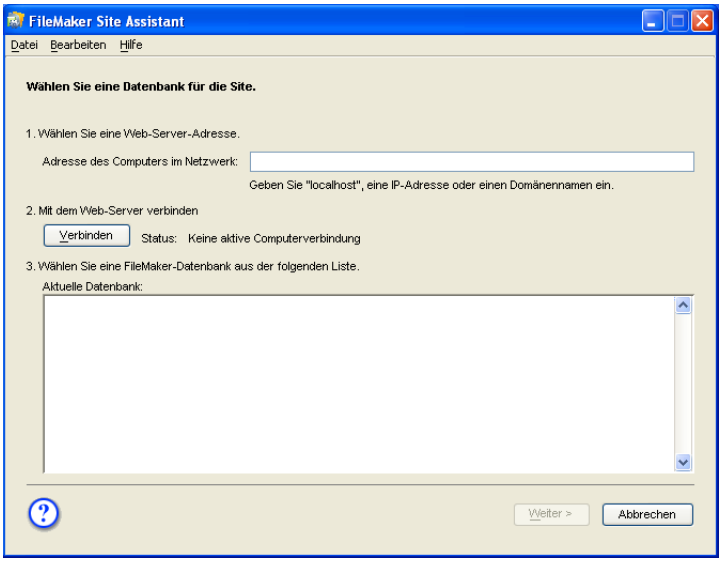
- Öffnen Sie die Datenbank mit FileMaker Server und geben Sie sie frei. Weitere Informationen finden Sie in der FileMaker Server Administration Hilfe
- Stellen Sie sicher, dass Web-Server und Web Publishing Engine laufen.
- Aktivieren Sie XSLT Publishing in der Web Publishing Engine, um die XSLT-Stylesheets zu verwenden und zu testen. Weitere Informationen erhalten Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.

Starten des Site Assistant

Um den Site Assistant zu starten, führen Sie eine der folgenden Aktionen aus:

- Doppelklicken Sie auf das FileMaker Site Assistant-Programmsymbol.

- Windows: Klicken Sie auf Start und wählen Sie im Menü Programme die Option FileMaker Site Assistant.


FileMaker Site Assistant
Datei Bearbeiten Hilfe
Wählen Sie eine Datenbank für die Site.
1. Wählen Sie eine Web-Server-Adresse.
Adresse des Computers im Netzwerk:
Geben Sie "localhost", eine IP-Adresse oder einen Domänennamen ein.
2. Mit dem Web-Server verbinden
Verbinden
Status: Keine aktive Computerverbindung
3. Wählen Sie eine FileMaker-Datenbank aus der folgenden Liste.
Aktuelle Datenbank:
Weiter >
Abbrechen

FileMaker Site Assistant

Verwenden des Site Assistant

Details und umfassende schrittweise Anweisungen für die Verwendung des Site Assistant finden Sie in der Site Assistant Hilfe. Informationen über die Verwendung der vom Site Assistant generierten XSLT-Stylesheets finden Sie unter „Verwenden von FileMaker XSLT-Stylesheets in einer Website oder einem Programm“ auf Seite 54.

Wichtig Wenn Sie bei Verwendung von Site Assistant eine Datenbank wählen, die mehrere Tabellen enthält, wählen Sie unbedingt Layouts, die mit dieser Tabelle verbunden sind, sonst liefert die generierte Site unerwartete Ergebnisse. Eine Datenbank könnte z. B. die Tabelle „Produkte“ und die Tabelle „Kunden“ enthalten. Wenn Sie die Layouts für eine Seite zum Suchen, eine Seite zur Datensatzbearbeitung und eine Seite zum Hinzufügen von Datensätzen wählen, müssen diese Layouts unbedingt mit derselben Tabelle verknüpft sein.

Erläuterung der vom Site Assistant generierten XSLT-Stylesheets

Die XSL-Stylesheets, die der Site Assistant generiert, umfassen mehrere FileMaker-spezifische Verarbeitungsanweisungen, Elemente und Parameter. Einige Beispiele dazu:

- Die Verarbeitungsanweisung <?xslt-cwp-query params="query string-fragment"?> gibt die zu verwendende XML-Grammatik an und definiert den Namen der im Site Assistant gewählten Datenbank statisch. Weitere Informationen finden Sie unter „Verwenden von statisch definierten Query-Befehlen und Query-Parametern“ auf Seite 62.
- Das Element <xsl:param name="request-query"/> wird verwendet, um auf Query-Informationen in einer Abfrage oder in HTML-Formulardaten zuzugreifen. Mithilfe dieses Elements kann z. B. in den Site Assistant-Stylesheets auf die aktuelle Abfrage-Query-Information zugegriffen werden, um die aktuelle Position in einer Datensatz-Ergebnismenge zu bestimmen und Verknüpfungen zum vorherigen und nächsten Datensatz zu erstellen. Weitere Informationen finden Sie unter „Verwenden der Query-Information in einer Abfrage“ auf Seite 66.
- Das Element <xsl:param name="authenticated-xml-base-uri"/>, das nicht immer eingeschlossen ist, wird verwendet, um auf den authentifizierten Basis-URI in einer Abfrage zuzugreifen, wenn in der Abfrage mehr XML-Daten benötigt werden. Weitere Informationen finden Sie unter „Verwenden des Parameters „authenticated base URI“ auf Seite 68.

Der Site Assistant generiert auch das Stylesheet utilities.xsl zur Definition von Fehlern und gängigen XSLT-Vorlagen, die von mehreren Site Assistant-Stylesheets aufgerufen werden.

Informationen über andere Abschnitte der Site Assistant-Stylesheets finden Sie unter Kapitel 5, „Entwickeln von FileMaker XSLT-Stylesheets“.

Verwenden von FileMaker CDML Converter

In FileMaker Server 7 Advanced wurde CDML (eine FileMaker-eigene Auszeichnungssprache) durch XSLT als Custom Web Publishing-Sprache für FileMaker-Datenbanken ersetzt. FileMaker unterstützt CDML nicht mehr. Um Websites von CDML-Dateien in FileMaker XSLT-Stylesheets umzuwandeln, verwenden Sie FileMaker CDML Converter.

Erläuterung von FileMaker CDML Converter

FileMaker CDML Converter ist ein Programm, das bestehende Dateien im CDML-Format in XSLT-Stylesheets umwandelt, die mit Custom Web Publishing mit XSLT kompatibel sind. Das ist ein nützliches Tool für den Prozess der CDML-Website-Migration und zum Kennenlernen des Aufbaus von FileMaker XSLT-Stylesheets.

Vor der Verwendung von CDML Converter wird empfohlen, die Dateien in CDML-Format aus der Produktionsumgebung in ein temporäres Arbeitsverzeichnis zu kopieren. Nach der Verwendung von CDML Converter müssen Sie die generierten Stylesheets und das Konvertierungsprotokoll überprüfen. In einigen Fällen sind die konvertierten XSLT-Stylesheets ohne weitere Änderung mit FileMaker Server einsatzbereit. In anderen Fällen kann der CDML Converter eventuell nicht alle CDML-Tags in XSLT-Stylesheets konvertieren und Sie müssen manuell einige Änderungen an den konvertierten XSLT-Stylesheets vornehmen. Sie können Ihr eigenes Erstellungswerkzeug für XSLT-Stylesheets oder Textverarbeitungswerkzeuge verwenden, um die Stylesheets je nach Bedarf zu ändern. Einige Änderungen an der Datenbank sind eventuell auch erforderlich, wenn in Datensätzen auf CDML-Dateien verwiesen wird. Dies liegt daran, dass die konvertierten XSLT-Stylesheets die Dateinamenerweiterung .xsl verwenden.

Sie können mit CDML Converter nur funktionierende CDML-Dateien in neue Stylesheets konvertieren. Sie können mit ihm keine konvertierten XSLT-Stylesheets bearbeiten. Für effektiven Einsatz von CDML Converter und Umwandlung der CDML-Website in XSLT sollten Sie ein erfahrener CDML-Entwickler und außerdem mit Custom Web Publishing mit XSLT vertraut sein.

Hinweis Die generierten XSLT-Stylesheets wandeln FileMaker-Daten auf der Basis der XML-Grammatik fmresultset in HTML-Seiten um.

Installieren von CDML Converter

Informationen über die Installation von CDML Converter finden Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.

Starten und Verwenden von CDML Converter

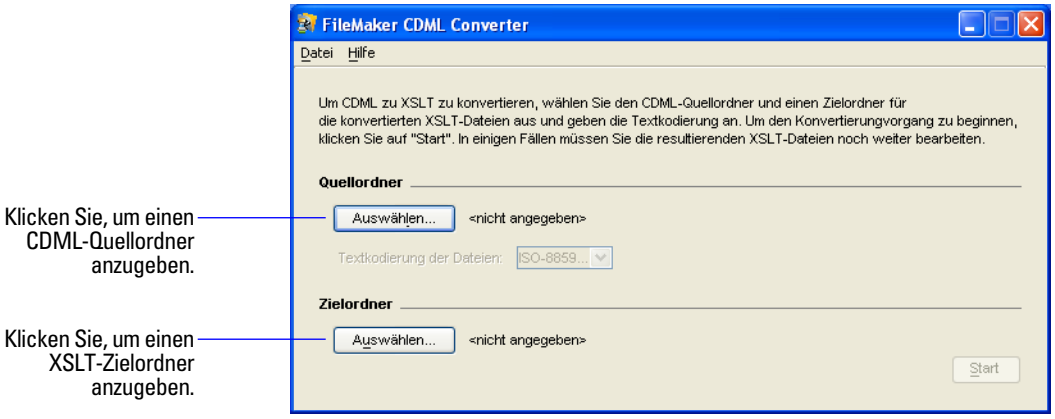
Wichtig Es wird empfohlen, zunächst alle Dateien aus dem Ordner cdml_format_files (falls verwendet) und alle Dateien aus Ihrem FileMaker Pro Web-Ordner in einen temporären Arbeitsordner zu kopieren und die Konvertierung an den kopierten Dateien auszuführen.

Hinweis Beim Konvertieren einer Quelldatei in einem XSLT-Stylesheet behält der CDML Converter den Dateinamen bei, ändert jedoch die Dateinamenerweiterung in .xsl. Wenn der Quellordner mehrere Dateien mit demselben Namen und unterschiedlichen Erweiterungen enthält, z. B. Datei.html und Datei.cdml, tritt bei der Konvertierung ein Problem mit dem Überschreiben der Dateien auf. Beispielsweise konvertiert der CDML Converter die Datei.html in Datei.xsl und versucht dann, Datei.cdml in Datei.xsl umzuwandeln. Da Datei.xsl bereits existiert, öffnet der CDML Converter ein Dialogfeld, in dem Sie bestätigen können, dass Sie die Datei überschreiben wollen. Um dieses Problem zu vermeiden, stellen Sie vor der Konvertierung sicher, dass die Dateien im Quellordner eindeutige Namen haben. Wenn Sie Dateinamen ändern, achten Sie darauf, dass Sie vor der Konvertierung auch alle etwaigen Verweise auf diese Dateien anpassen.

So starten und verwenden Sie den CDML Converter:

1. Um den CDML Converter zu starten, führen Sie eine der folgenden Aktionen aus:
   - Doppelklicken Sie auf das FileMaker CDML Converter-Programmsymbol.

   - Windows: Klicken Sie auf Start und wählen Sie im Menü Programme die Option FileMaker CDML Converter.


FileMaker CDML Converter
Datei Hilfe
Um CDML zu XSLT zu konvertieren, wählen Sie den CDML-Quellordner und einen Zielordner für die konvertierten XSLT-Dateien aus und geben die Textkodierung an. Um den Konvertierungvorgang zu beginnen, klicken Sie auf "Start". In einigen Fällen müssen Sie die resultierenden XSLT-Dateien noch weiter bearbeiten.
Quellordner
Auswählen...
<nicht angegeben>
Textkodierung der Dateien:
ISO-8859...
Zielordner
Auswählen...
<nicht angegeben>
Start
Klicken Sie, um einen CDML-Quellordner anzugeben.
Klicken Sie, um einen XSLT-Zielordner anzugeben.

FileMaker CDML Converter

2. Klicken Sie für Quellordner auf Auswählen, um den Ordner anzugeben, in dem die CDML-Quelldateien gespeichert sind.
3. Wählen Sie den Ordner aus, der die CDML-Quelldateien enthält, und klicken Sie dann auf Auswählen.
4. Wählen Sie für Textkodierung der Dateien die Kodierung der CDML-Quelldateien. Weitere Informationen finden Sie unter „Festlegen von Textkodierung für Abfragen“ auf Seite 63.

   Hinweis Alle Quellformatdateien in einer Konvertierungssitzung müssen dieselbe Textkodierung verwenden.
5. Klicken Sie für Zielordner auf Auswählen, um den Ordner anzugeben, in dem Sie die konvertierten XSLT-Stylesheets und andere Dateien aus der CDML-Website speichern wollen.

   Der CDML Converter erstellt im Zielordner eine Ordnerhierarchie und eine Gruppe von Dateien, die den Quelldateien entsprechen. Der CDML Converter kopiert Dateien unverändert aus dem Quellordner, wenn diese keine CDML-Konvertierung erfordern, wie z. B. GIF-Dateien in einem Unterordner mit Bildern. Die konvertierten Dateien haben denselben Namen, aber die Dateinamenerweiterung wird in .xsl geändert.
6. Wählen Sie den Ordner aus, in dem Sie die XSLT-Stylesheets auf Ihrer Festplatte speichern wollen, und klicken Sie dann auf Auswählen.
7. Klicken Sie auf Start.

   Der CDML Converter konvertiert die CDML-Dateien und speichert ein Konvertierungsprotokoll mit den konvertierten XSLT-Stylesheets im angegebenen Zielordner. Das Konvertierungsprotokoll wird auch im Fenster „CDML Converter“ angezeigt.
8. Klicken Sie auf Fertig.

Überprüfen und Korrigieren der vom CDML Converter generierten Stylesheets

Nach der Verwendung von CDML Converter können Sie im Konvertierungsprotokoll, das im Fenster „CDML Converter“ angezeigt wird, oder durch Öffnen des Konvertierungsprotokolls im Zielordner Fehlerinformationen über den Konvertierungsvorgang nachsehen. Die Konvertierungsprotokolldatei heißt cdml2xsl_<DatumZeit>.log, wobei <DatumZeit> das Datum und die Uhrzeit des Konvertierungsbeginns angibt.

Wenn bei der Konvertierung eine Warnung oder ein Fehler auftritt, fügt der CDML Converter dem Konvertierungsprotokoll eine Meldung hinzu und fügt in das konvertierte XSLT-Stylesheet ein XSLT-Kommentar-Tag mit einer Beschreibung des Problems ein. Die Fehler und Warnungen werden durch eines der folgenden XSLT-Kommentar-Tags im konvertierten XSLT-Stylesheet angezeigt:

<!—CDML CONVERTER-FEHLER: <Beschreibung des Fehlers> -->

<!—CDML CONVERTER-WARNUNG: <Beschreibung der Warnung> -->

Der CDML Converter verwendet einen angegebenen Satz von Zuordnungsregeln für CDML zu XSLT, um die CDML-Dateien zu konvertieren. Alle Konvertierungsfehler beruhen auf Situationen, in denen der CDML Converter die korrekte CDML-XSLT-Zuordnung nicht automatisch bestimmen konnte. Für alle Konvertierungsfehler müssen Sie die Ursache entweder manuell in der CDML-Datei korrigieren und die Datei dann erneut konvertieren oder die Probleme mithilfe eines Texteditors oder eines XSLT-Stylesheet-Editors manuell in den konvertierten XSLT-Stylesheets beheben.

Einige übliche Arten von Konvertierungsfehlern:

- Ein CDML-Tag wird in Custom Web Publishing mit XSLT nicht unterstützt. Beispielsweise wird das CDML-Aktions-Tag -dbclose nicht unterstützt. Um diese Art von Fehler zu beheben, müssen Sie entweder das CDML-Tag in ein unterstütztes Tag ändern oder die Funktionalität aus der CDML-Datei entfernen oder die Logik im XSLT-Stylesheet korrigieren.
- Ein CDML-Tag wird nicht erkannt. Ein CDML-Tag könnte falsch geschrieben sein. Um diese Art von Fehler zu beheben, müssen Sie entweder die Syntax des CDML-Tags entsprechend ändern, das Tag entfernen oder die Logik im XSLT-Stylesheet korrigieren.
- Ein CDML-Parameter wird nicht erkannt. Beispielsweise wird der CDML-Parameter „List“ im CDML-Tag [FMP-WerteListe: Field Name, List=WerteListenName] in XSLT-Stylesheets nicht unterstützt. Um diese Art von Fehler zu beheben, müssen Sie entweder die Syntax des CDML-Parameters entsprechend ändern, den Parameter entfernen oder die Logik im XSLT-Stylesheet korrigieren.

Weitere Informationen über die CDML-XSLT-Zuordnungsregeln und andere Situationen, in denen Sie XSLT-Anweisungen wegen nicht möglicher CDML-Tag-Konvertierung manuell ändern müssen, finden Sie in Anhang C, „Konvertieren von CDML-Lösungen in FileMaker XSLT“.

Hinweise

- Wenn mehrere Query-Parameter aus Datenbankfeldern auf CDML-Dateien verweisen, müssen Sie die Verweise in der Datenbank manuell aktualisieren. Wenn solche Verweise in konvertierten XSLT-Stylesheets gefunden werden, wird ein Hinweis in das XSLT-Stylesheet eingefügt, der Sie daran erinnert, die Datenbank entsprechend zu ändern.
- Wenn die konvertierten XSLT-Stylesheets eine FileMaker-Erweiterungsfunktion für Datum oder Uhrzeit enthalten, z. B. die Funktion fmxslt:get_date(), verwendet die Funktion die „fm“-Formate für Zeichenfolgen. Die „fm“-Formate sind MM/tt/jjjj für Datum, HH:mm:ss für Zeit und MM/tt/jjjj HH:mm:ss für Zeitstempel. Weitere Informationen finden Sie unter „Verwenden der Datums-, Zeit- und Tages-Erweiterungsfunktionen“ auf Seite 81. Nach der Konvertierung müssen Sie Datums- und Zeitformatstrings, die an diese Funktionen übergeben werden, manuell ändern, z. B. Monat/Tag/Jahr in Tag.Monat.Jahr.

Verwenden der vom CDML Converter generierten Stylesheets

Nach dem Beheben der Fehler in den konvertierten XSLT-Stylesheets können Sie sie mit der Web Publishing Engine benutzen. Weitere Informationen finden Sie im nächsten Abschnitt, „Verwenden von FileMaker XSLT-Stylesheets in einer Website oder einem Programm“.

Testen der vom CDML Converter generierten Stylesheets

Sie sollten die konvertierten XSLT-Stylesheets gründlich testen, bevor Sie sie in einer Produktionsumgebung verwenden. Weitere Informationen finden Sie unter Kapitel 6, „Testen und Überwachen einer Site“.

Verwenden von FileMaker XSLT-Stylesheets in einer Website oder einem Programm

Unabhängig davon, ob Sie XSLT-Stylesheets mithilfe von Site Assistant oder CDML Converter generiert oder von Grund auf neu entwickelt haben, sind die Schritte für die Benutzung in einer Website oder einem Programm mit der Web Publishing Engine gleich.

So verwenden Sie FileMaker XSLT-Stylesheets in einer Website oder einem Programm:

1. Kopieren oder platzieren Sie die XSLT-Stylesheets im Ordner xslt-template-files, der sich im Ordner Web Publishing im Ordner FileMaker Server auf dem Host befindet, auf dem die Web Publishing Engine installiert ist.

   Sie können die Stylesheets auch in einem optionalen Ordner oder einer Ordnerhierarchie innerhalb des Ordners xslt-template-files platzieren.

2. Wenn Ihre XSLT-Stylesheets auf statische Dateien verweisen, z. B. statische Bilder oder HTML-Dateien, platzieren Sie die statischen Dateien mit ihrer originalen Ordnerhierarchie innerhalb des Root-Ordners auf dem Web-Server. Stellen Sie sicher, dass der relative Pfad beibehalten wird.

   Nehmen Sie z. B. an, ein XSLT-Stylesheet verweist auf die Bilddatei logo.jpg und verwendet dafür das HTML-Tag <img src="logo.jpg">. Die Datei logo.jpg muss sich auf dem Web-Server an der folgenden Position befinden:

   <root folder>/fmi/xsl/logo.jpg

3. Falls ein Datenbank-Medienfeld einen Dateiverweis anstelle einer tatsächlichen Datei speichert, muss das entsprechende Medienobjekt im Ordner FileMaker Pro Web gespeichert sein, wenn der Datensatz erstellt oder bearbeitet wird, und dann in einen Ordner mit dem gleichen relativen Speicherort im Root-Ordner der Web-Server-Software kopiert oder verschoben werden. Weitere Informationen finden Sie unter „Veröffentlichen des Inhalts von Medienfeldern im Web“ auf Seite 23.

   Hinweis Falls das Medienfeld die tatsächlichen Dateien in der FileMaker-Datenbank speichert, ist mit dem Inhalt des Medienfelds keine Aktion erforderlich, wenn die Datenbank richtig auf dem FileMaker Server freigegeben und zugänglich ist.

4. Um ein XSLT-Stylesheet abzufragen und zu verarbeiten, verwenden Sie die folgende URL-Syntax:

   <scheme>://<host>[:<port>]/fmi/xsl/<folder>/<stylesheet>.xsl[?<query string>]

   Weitere Informationen finden Sie unter „Erläuterung der URL-Syntax für FileMaker XSLT-Stylesheets“ auf Seite 58.

Hinweis Für Websites ist es praktisch, ein XSLT-Stylesheet als Homepage zu verwenden, die nicht verlangt, dass Benutzer für den Zugriff einen Query-String eingeben. Der Site Assistant kann eine Datei home.xsl erzeugen, die keinen Query-String verlangt, da sie die Verarbeitungsanweisung <?xslt-cwp-query?> verwendet. Wenn Sie z. B. Ihre Stylesheets (einschließlich eines home.xsl-Stylesheets) in den Ordner my_templates im Ordner xslt-template-files kopiert haben, können Sie folgenden URL verwenden, um die Stylesheets anzufordern und zu verarbeiten:

http://192.168.123.101/fmi/xsl/my_templates/home.xsl

Wichtig Die Web Publishing Engine erlaubt Web-Benutzern nicht, den Quelltext der XSLT-Stylesheets anzuzeigen, die im Ordner xslt-template-files installiert sind. Wenn Web-Benutzer eine Anforderung für die Verarbeitung eines Stylesheets senden, sendet die Web Publishing Engine nur das Ergebnis der Stylesheet-Umwandlung an den Webbrowser bzw. das Programm.

Fehlerbehebung in XSLT-Stylesheets

Wenn Probleme beim Verwenden der XSLT-Stylesheets auftreten, überprüfen Sie das Folgende:

- Die erweiterten Zugriffsrechte in der Datenbank müssen für Custom Web Publishing mit XSLT eingestellt und einem Benutzerkonto zugewiesen sein. Weitere Informationen finden Sie unter „Aktivieren von Instant Web Publishing in einer Datenbank“ auf Seite 21.
- Die Datenbank muss über FileMaker Server bereitgestellt und geöffnet sein. Weitere Informationen finden Sie in der FileMaker Server Administration Hilfe
- Prüfen Sie, ob der verwendete Datenbankkontoname und ggf. Ihr Passwort korrekt sind.
- Stellen Sie sicher, dass Web-Server und Web Publishing Engine laufen.
- XSLT-Publishing muss in der Web Publishing Engine aktiviert sein. Weitere Informationen erhalten Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.

Kapitel 5
Entwickeln von FileMaker XSLT-Stylesheets

Dieses Kapitel enthält Informationen über den Aufbau von FileMaker XSLT-Stylesheets und die Verwendung der FileMaker XSLT-Erweiterungsfunktionen.

Verwenden von XSLT-Stylesheets mit der Web Publishing Engine

Beachten Sie beim Entwickeln und Verwenden von XSLT-Stylesheets für die Abfrage von FileMaker XML-Daten über die Web Publishing Engine die folgenden Punkte:

- Um ein XSLT-Stylesheet mit der Web Publishing Engine zu verwenden, müssen Sie den Namen eines XSLT-Stylesheets in einem URL angeben. Wenn Sie kein Stylesheet angeben oder die Web Publishing Engine das Stylesheet nicht findet oder analysiert („parst“), zeigt die Web Publishing Engine eine Fehlerseite an. Weitere Informationen finden Sie unter „Erläuterung der URL-Syntax für FileMaker XSLT-Stylesheets auf Seite“ 58.
- Der Stylesheet-Dateiname und der Name des Ordners, in dem das Stylesheet gespeichert ist, muss URL-kodiertes UTF-8 sein. Falls Ihr Stylesheet mit älteren Webbrowsern kompatibel sein muss, beschränken Sie die Namen auf ASCII-Zeichen.
- Sie müssen die zu verwendende FileMaker XML-Grammatik angeben, entweder als Query-Parameter im URL oder als statisch definierten Query-Parameter in der Verarbeitungsanweisung <?xslt-cwp-query?>. Wenn Sie keine XML-Grammatik angeben, zeigt die Web Publishing Engine einen Fehler an. Weitere Informationen finden Sie unter „Angeben einer XML-Grammatik für ein FileMaker XSLT-Stylesheet auf Seite“ 60.
- Sie können die Query-Parameter zur Identifikation der angeforderten FileMaker XML-Daten im URL oder als statisch definierten Query-Parameter in der Verarbeitungsanweisung <?xslt-cwp-query?> angeben. Weitere Informationen hierzu finden Sie unter „Erläuterung der URL-Syntax für FileMaker XSLT-Stylesheets“ auf Seite 58 und „Verwenden von statisch definierten Query-Befehlen und Query-Parametern auf Seite“ 62.
- Optional können Sie die Textkodierung einer XSLT-Abfrage mithilfe des Query-Parameters –encoding angeben. Wenn Sie keine Kodierung angeben, verwendet die Web Publishing Engine ihre Standardkodierungseinstellung für Abfragen. Weitere Informationen finden Sie unter „Festlegen von Textkodierung für Abfragen auf Seite“ 63.
- Sie können optional eine Ausgabemethode über das method-Attribut des Elements <xsl:output> angeben. Wenn Sie keine Ausgabemethode angeben, verwendet die Web Publishing Engine HTML als Ausgabe. Sie können optional die Kodierung der Ausgabeseite über das encoding-Attribut des Elements <xsl:output> angeben. Wenn Sie keine Kodierung angeben, verwendet die Web Publishing Engine ihre Standardkodierungseinstellung für Ausgabeseiten. Weitere Informationen finden Sie unter „Angeben einer Ausgabemethode und -kodierung auf Seite“ 64.
- Sie können optional für E-Mail-Nachrichten, die von der Web Publishing Engine gesendet werden, die Textkodierung über einen Funktionsparameter für die Erweiterungsfunktion fmxslt:send_email() angeben. Weitere Informationen finden Sie unter „Senden von E-Mail-Nachrichten von der Web Publishing Engine auf Seite“ 74.

Den Aufbau einer Abfrage beginnt die Web Publishing Engine mit einem beliebigen Query-Befehl und Query-Parametern, die statisch in der optionalen Verarbeitungsanweisung <?xslt-cwp-query?> gespeichert sind. Der statisch definierte Query-Befehl und seine Parameter werden zur Basisabfrage. Die Verarbeitungsanweisung <?xslt-cwp-query?> ist in einem Stylesheet nicht erforderlich, aber ihre Basisabfrage hat Vorrang vor entsprechenden Query-Befehlen oder -Parametern, die im URL-Query-String angegeben sind. Die Web Publishing Engine fügt dann der Basisabfrage im URL-Query-String einen Query-Befehl oder zusätzliche Parameter hinzu, die *nicht* in der Verarbeitungsanweisung <?xslt-cwp-query?> definiert sind. Die Web Publishing Engine bezieht anhand dieser Abfrage die FileMaker XML-Daten und gibt sie in der angegebenen Ausgabemethode bzw. als HTML an Ihren Webbrowser oder Ihr Programm zurück.

Referenz der FileMaker XSLT-Erweiterungsfunktionen

Die FileMaker Server Web Publishing-CD enthält eine FileMaker-Datenbank mit dem Namen XSLT Reference.fp7, die kurze Beschreibungen und Beispiele für jede FileMaker XSLT-Erweiterungsfunktion bietet. Sie können diese Referenz der FileMaker XSLT-Erweiterungsfunktionen im Ordner Custom Web Publishing Reference auf der FileMaker Server Web Publishing-CD öffnen.

Erläuterung der URL-Syntax für FileMaker XSLT-Stylesheets

Die URL-Syntax für die Verwendung von FileMaker XSLT-Stylesheets mit der Web Publishing Engine:

<schema>://<host>[:<port>]/fmi/xsl/[<pfad>/]<stylesheet.xsl>[?<query_string>]

Dabei gilt:

- <schema> kann das HTTP- oder HTTPS-Protokoll sein.
- <host> ist die IP-Adresse oder der Domänenname des Hosts, auf dem der Web-Server installiert ist.
- <port> ist optional und gibt den Port an, den der Web-Server überwacht. Ohne Portangabe wird der Standard-Port des Protokolls (Port 80 für HTTP oder Port 443 für HTTPS) verwendet.
- <path> ist optional und gibt den/die Ordner innerhalb des Ordners xslt-template-files an, in dem sich das XSLT-Stylesheet befindet.
- <stylesheet.xsl> ist der Dateiname des XSLT-Stylesheets.
- <query_string> kann eine Kombination aus einem Query-Befehl und einem oder mehreren Query-Parametern für Custom Web Publishing mit XSLT sein. Weitere Informationen finden Sie unter „Verwenden von Query-Strings in FileMaker XSLT-Stylesheets“ auf Seite 60, und in Anhang A, „Gültige Namen in Query-Strings“. Wenn das angegebene Stylesheet eine <?xslt-cwp-query?>-Verarbeitungsanweisung enthält, haben der statisch definierte Query-Befehl und die Parameter Vorrang vor anderen Query-Befehlen oder Parametern im URL-Query-String. Weitere Informationen finden Sie unter „Verwenden von statisch definierten Query-Befehlen und Query-Parametern auf Seite“ 62.

Hinweis Die URL-Syntax einschließlich der Namen von Query-Befehl und -Parametern unterscheidet Groß- und Kleinschreibung, mit Ausnahme von Teilen des Query-Strings. Der URL ist vorwiegend in Kleinbuchstaben mit Ausnahme der Grammatiknamen in Großbuchstaben: FMPXMLRESULT und FMPXMLLAYOUT. Informationen über die Regeln für Groß-/Kleinschreibung im Query-String finden Sie unter „Richtlinien für die Verwendung von Query-Befehlen und -Parametern auf Seite“ 96.

Beispiel eines URL für die Verwendung eines FileMaker XSLT-Stylesheets mit der Web Publishing Engine:

http://192.168.123.101/fmi/xsl/my_template/my_stylesheet.xsl?–grammar=fmresultset&–db=mydatabase&–lay=mylayout&–findall

Erläuterung der URL-Syntax für FileMaker-Medienobjekte in XSLT-Lösungen

In einem generierten XML-Dokument für eine XSLT-Lösung unterscheidet sich die Syntax für den Verweis auf ein Medienobjekt für Medienfelder, die das tatsächliche Objekt in der Datenbank speichern, und für Medienfelder, die einen Verweis auf das Objekt speichern.

- Wenn ein Medienfeld das tatsächliche Objekt in der Datenbank speichert, dann verwendet das <data>-Element des Medienfelds die folgende URL-Syntax für den Verweis auf das Objekt:

  <data>/fmi/xsl/cnt/data.<extension>?<query string></data>.

  Dabei gilt: <extension> ist die Dateinamenerweiterung zur Identifizierung des Objekttyps wie beispielsweise .jpg oder .mov. Informationen über <query string> finden Sie im vorherigen Abschnitt, „Erläuterung der URL-Syntax für FileMaker XSLT-Stylesheets“.

  Beispiel:

  <data>/fmi/xsl/cnt/data.jpg?–db=products&–lay=sales&–field=product_image(1)&–recid=2</data>

  Hinweis In der generierten XML für ein Medienfeld ist der Wert für den Query-Parameter –field ein vollständig qualifizierter Feldname. Die Zahl in Klammern gibt die Wiederholungsnummer für das Medienfeld an und wird für Wiederhol- und für Einzelfelder generiert. Weitere Informationen finden Sie unter „Erläuterung der Syntax für einen voll qualifizierten Feldnamen auf Seite“ 97.

  Verwenden Sie zum Abrufen der Mediendaten aus der Datenbank die folgende Syntax:

  <schema>://<host>[:<port>]/fmi/xsl/cnt/data.<erweiterung>?<query_string>

  Informationen über <schema>, <host> oder <port> finden Sie im vorherigen Abschnitt, „Erläuterung der URL-Syntax für FileMaker XSLT-Stylesheets“.

  Beispiel:

  http://www.company.com/fmi/xsl/cnt/data.jpg?–db=products&–lay=sales&–field=product_image(1)&–recid=2

- Wenn ein Medienfeld einen Dateiverweis anstelle eines tatsächlichen Objekts speichert, enthält das <data>-Element des Medienfelds einen relativen Pfad, der auf das Objekt verweist. Wenn sich z. B. logo.jpg im Web-Ordner innerhalb des FileMaker Pro-Ordners befindet, lautet das <data>-Element des Medienfelds:

  <data>/images/logo.jpg</data>

  Hinweis Das Medienobjekt, auf das verwiesen wird, muss im Ordner FileMaker Pro Web gespeichert sein, wenn der Datensatz erstellt oder bearbeitet wird, und dann in einen Ordner mit dem gleichen relativen Speicherort im Root-Ordner der Web-Server-Software kopiert oder verschoben werden. Weitere Informationen finden Sie unter „Veröffentlichen des Inhalts von Medienfeldern im Web auf Seite“ 23.

- Für ein leeres Medienfeld ist auch das <data>-Element leer.

Verwenden von Query-Strings in FileMaker XSLT-Stylesheets

Beim Verwenden eines Query-Strings in einem URL oder der Verarbeitungsanweisung <?xslt-cwp-query?> in einem FileMaker XSLT-Stylesheet können Sie beliebige Query-Befehle und -Parameter einschließen, die zur Abfrage von XML-Daten aus einer FileMaker-Datenbank definiert sind. Weitere Informationen finden Sie unter „Abfragen von XML-Daten mithilfe von FileMaker-Query-Strings auf Seite“ 40.

Sie können auch den folgenden Query-Befehl und Parameter benutzen, die nur zur Verwendung mit FileMaker XSLT-Stylesheets definiert sind.

XSLT-Query-Befehl oder -Parameternamen	Für	Kommentar
–grammar	Angeben der XML-Grammatik für XSLT-CWP-Abfragen oder für XSLT-Stylesheets. Weitere Informationen finden Sie im nächsten Abschnitt, „Angeben einer XML-Grammatik für ein FileMaker XSLT-Stylesheet“.	Dieser Query-Parameter ist in allen XSLT-Abfragen erforderlich.
–encoding	Angeben der Textkodierung für eine Abfrage. Weitere Informationen finden Sie unter „Festlegen von Textkodierung für Abfragen auf Seite“ 63.	Dieser Query-Parameter ist in allen XSLT-Abfragen optional.
–process	Verarbeitung eines Stylesheets ohne Datenabfrage. Weitere Informationen finden Sie unter „Verarbeitung von XSLT-Abfragen, die nicht FileMaker Server abfragen auf Seite“ 64.	Dieser Query-Befehl erfordert den Query-Parameter –grammar.
–token	Weitergeben von Werten zwischen Seiten, ohne Sitzungen oder Cookies zu verwenden. Weitere Informationen finden Sie unter „Verwendung von Tokens zur Übergabe von Informationen zwischen Stylesheets auf Seite“ 65.	Dieser Query-Parameter ist in allen XSLT-Abfragen optional.

Angeben einer XML-Grammatik für ein FileMaker XSLT-Stylesheet

Die empfohlene XML-Grammatik für Custom Web Publishing mit XSLT ist die Grammatik fmresultset, die zur bequemen Benutzung mit XSLT entwickelt wurde. Weitere Informationen finden Sie unter „Verwenden der fmresultset-Grammatik auf Seite“ 33. Sie können auch die älteren Grammatiken FMPXMLRESULT oder FMPXMLLAYOUT verwenden. Für den Zugriff auf Wertelisten und Feldanzeigedaten in Layouts müssen Sie die FMPXMLLAYOUT-Grammatik verwenden. Weitere Informationen finden Sie unter „Verwenden anderer FileMaker XML-Grammatiken auf Seite“ 36. Die Grammatik FMPDSORESULT kann nicht mit Custom Web Publishing mit XSLT verwendet werden.

Um die Grammatik für ein FileMaker XSLT-Stylesheet anzugeben, verwenden Sie den Query-Parameter –grammar in einem URL oder den statisch definierten Query-Parameter in der Verarbeitungsanweisung <?xslt-cwp-query?>.

Beispiel:

http://192.168.123.101/fmi/xsl/my_template/my_stylesheet.xsl?–grammar=fmresultset&–db=mydatabase&–lay=mylayout&–findall

Oder

<?xslt-cwp-query params="–grammar=fmresultset&–db=mydatabase&–lay=mylayout&–findall"?>

Wichtig Wenn Sie keine XML-Grammatik für ein FileMaker XSLT-Stylesheet angeben, wird der Fehler QUERY -ER0001 angezeigt. Weitere Informationen finden Sie unter Anhang B, „Fehlercodes für Custom Web Publishing“.

Erläuterung von Namespaces und Präfixen für FileMaker XSLT-Stylesheets

Mithilfe eindeutiger XSLT-Namespaces können Sie XSLT-Tags der richtigen Anwendung zuordnen. Deklarieren Sie im Element <xsl:stylesheet> am Beginn aller FileMaker XSLT-Stylesheets die Namespaces für die FileMaker XSLT-Erweiterungsfunktionen und die jeweiligen Grammatiken, die Sie in dem Stylesheet verwenden.

Verwendung	Namespace deklarieren	Präfix verwenden
fmresultset-XML-Grammatik	xmlns:fmrs="http://www.filemaker.com/xml/fmresultset"	fmrs
FMPXMLRESULT-Grammatik	xmlns:fmp="http://www.filemaker.com/fmpxmlresult"	fmp
FMPXMLLAYOUT-Grammatik	xmlns:fml="http://www.filemaker.com/fmpxmllayout"	fml
Für die Query-XML-Grammatik	xmlns:fmq="http://www.filemaker.com/xml/query"	fmq
Für die FileMaker XSLT-Erweiterungsfunktionen	xmlns:fmxslt="xalan://com.fmi.xslt.ExtensionFunctions"	fmxslt

Sie müssen auch den folgenden erforderlichen Namespace in jedem FileMaker XSLT-Stylesheet deklarieren:

xmlns:xsl="http://www.w3.org/1999/XSL/Transform"

Beispiel für Deklarationen von Namespaces:

```
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fmrs="http://www.filemaker.com/xml/fmresultset"
    xmlns:fml="http://www.filemaker.com/fmpxmllayout"
    xmlns:fmq="http://www.filemaker.com/xml/query"
    xmlns:fmxslt="xalan://com.fmi.xslt.ExtensionFunctions"
    exclude-result-prefixes="xsl fmrs fmq fml fmxslt">
```

Verwenden von statisch definierten Query-Befehlen und Query-Parametern

Sie können mit Ihrem FileMaker XSLT-Stylesheet die unbefugte Verwendung von Query-Befehlen und Query-Parametern verhindern, indem Sie die zu verwendenden Query-Befehle und -Parameter beim Abfragen von XML-Daten statisch definieren. Wenn Query-Befehle und -Parameter statisch in einem Stylesheet definiert sind (was nicht erforderlich ist), haben sie Vorrang vor etwaigen übereinstimmenden Query-Befehlen oder -Parametern, die ein URL-Query-String angibt.

Die Stylesheets, die von dem Tools Site Assistant und dem CDML Converter generiert werden, verwenden statisch definierte Query-Befehle und -Parameter. FileMaker empfiehlt die Verwendung statisch definierter Query-Befehle und -Parameter als beste Technik für eine größere Sicherheit Ihrer Lösung.

Um Query-Befehle und -Parameter statisch zu definieren, verwenden Sie die folgende Verarbeitungsanweisung am Beginn Ihres FileMaker XSLT-Stylesheets:

```
<?xslt-cwp-query params="query string-fragment"?>
```

Dabei gilt:

query string-fragment ist ein String mit den Name-Wert-Paaren in folgendem Format:

name=wert&name2=wert2....

Dabei gilt:

name ist ein String, der den Namen eines Query-Befehls, Query-Parameters oder Datenbankfelds angibt.

wert ist ein Stringwert beliebiger Länge. Verwenden Sie für Query-Parameter und Feldnamen den jeweiligen Wert, den Sie definieren wollen, z. B. –db=Produkte. Geben Sie für Query-Befehle kein =-Zeichen oder einen Wert nach dem Befehlsnamen wie –findall an. Weitere Informationen finden Sie unter Anhang A, „Gültige Namen in Query-Strings“.

Die im Fragment verwendeten Strings müssen URL-kodiert sein. Weitere Informationen finden Sie unter „Erläuterung von URL-Textkodierung auf Seite“ 31. Sie müssen dieselbe Zeichenkodierung verwenden, die das encoding-Attribut im Tag <xsl:output> angibt. Wenn keine Kodierung angegeben wird, verwendet die Web Publishing Engine ihre konfigurierte Standardkodierung.

Das Trennzeichen zwischen den beiden Name-Wert-Paaren muss ein kaufmännisches Und-Zeichen (&) sein.

Nehmen Sie z. B. an, Sie haben die folgende Verarbeitungsanweisung in einem Stylesheet mit dem Namen my_stylesheet.xsl verwendet:

```
<?xslt-cwp-query params="–db=products&–lay=sales&–grammar=fmresultset&productname=the%20item&–find"?>
```

Diese Beispiel-Verarbeitungsanweisung würde alle Abfragen an my_stylesheet.xsl zwingen, die fmresultset-Grammatik mit der Datenbank products und dem Layout sales zu verwenden und eine -find-Abfrage mit dem Wert the%20item im Feld „productname“.

Nun führt ein Client die folgende Abfrage mit my_stylesheet.xsl durch:

http://server.company.com/fmi/xsl/my_stylesheet.xsl?–lay=revenue&city=London&–edit

Die Web Publishing Engine verarbeitet in diesem Fall die folgende XML-Abfrage:

http://server.company.com/fmi/xml/fmresultset.xml?–db=products&–lay=sales&productname=the%20item&city=London&–find

Der statisch definierte Query-Befehl und seine Parameter setzen den Query-Parameter –lay=revenue und den Query-Befehl –edit außer Kraft, die der Client angegeben hat. Da das Feld „city“ in der Verarbeitungsanweisung nicht statisch definiert war, übernimmt die Web Publishing Engine dafür in der XML-Abfrage den Wert London, den der Client eingegeben hat.

Festlegen von Textkodierung für Abfragen

Die Web Publishing Engine führt die folgenden Schritte in der gezeigten Reihenfolge aus, bis sie die Kodierung einer XSLT-Abfrage ermittelt:

1. Prüfen, ob das charset-Attribut im Abfrage-Header „Content-Type“ gesetzt ist.
2. Prüfen, ob Sie eine Kodierung mit dem Query-Parameter –encoding angegeben haben. Sie können diesen Parameter in einem URL oder als statisch definierten Query-Parameter in der Verarbeitungsanweisung <?xslt-cwp-query?> angeben. Der Wert des –encoding-Parameters gibt die Kodierung an, die für die übrigen Parameter in der Abfrage verwendet wird. Die gültigen Werte für diesen Parameter werden in der folgenden Tabelle aufgelistet. Beispiel:

   http://192.168.123.101/fmi/xsl/template/my_stylesheet.xsl?–db=products–lay=sales&–grammar=fmresultset&–encoding=Shift_JIS&–findall
3. Verwenden der aktuellen Einstellung für die Standard-Textkodierungsoption der Abfrage- und Ausgabeseiten für die Web Publishing Engine. Direkt nach der Installation der Web Publishing Engine ist die Textkodierung für Abfragen standardmäßig auf UTF-8 eingestellt. Sie können die Textkodierungseinstellungen der Web Publishing Engine über die Administrationskonsole ändern. Weitere Informationen erhalten Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.

Nachdem die Web Publishing Engine die Kodierung ermittelt, wird diese Kodierung verwendet und es werden keine weiteren Schritte zur Ermittlung der Kodierung unternommen. Wenn z. B. das charset-Attribut im Abfrage-Header „Content-Type“ gesetzt ist, verwendet die Web Publishing Engine nicht den Wert des Query-Parameters –encoding.

Die Textkodierung, die über eine der obigen Methoden angegeben wird, muss eine der folgenden Kodierungen verwenden:

Kodierung	Beschreibung
US-ASCII	Der ASCII-Basiszeichensatz wird typischerweise für englische E-Mails mit Standardtext verwendet.
ISO-8859-1	Der Latin 1-Zeichensatz wird typischerweise für Web-Seiten und E-Mail-Nachrichten mit lateinischen Buchstaben aus dem oberen ASCII-Zeichenbereich verwendet.
ISO-8859-15	Der Latin 9-Zeichensatz, der beinahe mit dem Latin 1-Zeichensatz übereinstimmt, umfasst zusätzlich das Euro-Symbol €.
ISO-2022-JP	Die ISO Japanese-Kodierung wird typischerweise für japanische E-Mail-Nachrichten verwendet.
Shift_JIS	Diese Japanisch-Kodierung wird typischerweise für japanische Web-Seiten verwendet.
UTF-8	Die Acht-Bit-Kodierung von Unicode. Die Verwendung von UTF-8 für E-Mail-Nachrichten und Web-Seiten gewinnt an Beliebtheit, da gängige Browser und E-Mail-Clients diese Kodierung jetzt unterstützen. Da UTF-8 den kompletten Bereich an Unicode-Zeichen unterstützt, sind damit Seiten in jeder Sprache möglich.

Hinweise

- Direkt nach der Installation der Web Publishing Engine ist die Textkodierung für Ausgabeseiten standardmäßig auf UTF-8 eingestellt. Weitere Informationen finden Sie im nächsten Abschnitt, „Angeben einer Ausgabemethode und -kodierung“. Für E-Mail-Nachrichten verwendet die Web Publishing Engine anfangs für die Textkodierung ISO-8859-1 als Standardeinstellung. Sie können diese Einstellung über die Administrationskonsole ändern.
- Sie können auch die E-Mail-Nachrichtenkodierung mithilfe der Erweiterungsfunktion fmxslt:send_email(String smtpFields, String body, String encoding) festlegen. Weitere Informationen finden Sie unter „Senden von E-Mail-Nachrichten von der Web Publishing Engine auf Seite“ 74.

Angeben einer Ausgabemethode und -kodierung

Sie können eine Ausgabemethode und -kodierung der Ausgabeseiten mithilfe des method- und encoding-Attributs des Elements <xsl:output> angeben. Beide Attribute sind optional.

Das method-Attribut legt den Typ der Ausgabe fest, d. h. „html“, „text“ oder „xml“. Andere Methodentypen werden nicht unterstützt. Wenn Sie keine Methode angeben, verwendet die Web Publishing Engine „html“.

Das encoding-Attribut gibt die Kodierung von Ausgabeseiten an. Sie können eine beliebige Kodierung angeben, die in der Tabelle im vorherigen Abschnitt aufgeführt ist. Wenn Sie keine Kodierung angeben, verwendet die Web Publishing Engine ihre Standardkodierungseinstellung für Ausgabeseiten.

Beispiel:

```
<xsl:output method="html" encoding="ISO-8859-1"/>
```

Wenn Sie das Element <xsl:output> nicht in einem Stylesheet verwenden, gibt die Web Publishing Engine HTML-Seiten in der aktuellen Standardeinstellung für Textkodierung von Ausgabeseiten aus.

Erläuterung der Kodierung von XSLT-Stylesheets

Neben der Kodierung für Abfragen und Ausgabeseiten muss die Kodierung Ihrer XSLT-Stylesheets auch im encoding-Attribut der XML-Deklaration am Beginn des Stylesheets angegeben werden. Sie können eine beliebige Kodierung angeben, aufgeführt in der Tabelle auf Seite 63.

Beispielsweise gibt die folgende Deklaration UTF-8 als Kodierung des Stylesheets an:

```
<?xml version="1.0" encoding="UTF-8"?>
```

Wenn Sie keine Stylesheet-Kodierung angeben, nimmt die Web Publishing Engine UTF-8 als Kodierung an.

Verarbeitung von XSLT-Abfragen, die nicht FileMaker Server abfragen

Sie können mithilfe des Query-Befehls –process XSLT-Abfragen verarbeiten, wenn diese keine Daten aus der Datenbank benötigen oder wenn Ihr Stylesheet keine datenbankspezifischen Informationen wie Datensätze, Feldnamen oder Layoutnamen erfordert. Indem Sie in diesen Fällen den Befehl –process verwenden, können Sie die Arbeitslast von FileMaker Server verringern.

Mithilfe des Befehls –process können Sie z. B.:

- ein Stylesheet laden, das eine statische Seite generiert, wenn keine Datenbankinformation benötigt wird.
- ein Stylesheet laden, das einen neuen Datensatz erstellt, wenn das Stylesheet keine Datenbank- oder Layoutinformation wie eine Werteliste benötigt.

- eine Erweiterungsfunktion wie fmxslt:send_email() verwenden, die keine Daten aus der Datenbank benötigt.
- auf Informationen zugreifen, die in einer Sitzung gespeichert sind, wenn keine Datenbankinformation benötigt wird.

Der Befehl –process gibt ein XML-Dokument zurück, das Produktinformationen über die Web Publishing Engine enthält.

Der einzige erforderliche Parameter für den Befehl –process ist –grammar und Sie müssen die fmresultset- oder die FMPXMLRESULT-Grammatik verwenden.

Beispiel:

http://192.168.123.101/fmi/xsl/my_template/my_stylesheet.xsl?–grammar=fmresultset&–process

Verwendung von Tokens zur Übergabe von Informationen zwischen Stylesheets

Sie können den Query-Parameter –token in einem URL oder als statisch definierten Query-Befehl verwenden, um beliebige benutzerdefinierte Informationen ohne Sitzungen oder Cookies zwischen Stylesheets zu übergeben. Der Query-Parameter –token ist in allen Query-Befehlen optional.

Der benutzerdefinierte Parameterwert kann ein beliebiger URL-kodierter String sein. Beispiel:

http://192.168.123.101/fmi/xsl/template/my_stylesheet.xsl?–db=products&–lay=sales&–grammar=fmresultset&–token.D100=Pending&–findall

Weitere Informationen finden Sie unter „Query-Parameter –token.[string] (Werte zwischen XSLT-Stylesheets übergeben) auf Seite“ 112.

Wichtig Verwenden Sie den Query-Parameter –token nicht, um private Daten zu übergeben.

Um den Wert des Query-Parameters –token abzurufen, verwenden Sie die Anweisung <xsl:param name="request-query"/>. Weitere Informationen finden Sie unter „Verwenden der Query-Information in einer Abfrage auf Seite“ 66.

Verwenden der FileMaker XSLT-Erweiterungsfunktionen und -parameter

Die FileMaker XSLT-Erweiterungsfunktionen befinden sich laut Definition im fmxslt-Namespace. Stellen Sie sicher, dass Sie eine Deklaration des fmxslt-Namespace im Element <xsl:stylesheet> am Beginn Ihres XSLT-Stylesheets angeben. Weitere Informationen finden Sie unter „Erläuterung von Namespaces und Präfixen für FileMaker XSLT-Stylesheets auf Seite“ 61.

Die FileMaker XSLT-Erweiterungsfunktionen wurden so entwickelt, dass Sie sie in einem XSLT-Stylesheet verwenden können, indem Sie sie als Funktionsaufruf in einer XPath-Anweisung angeben. XPath-Anweisungen werden als Werte des select-Attributs und des test-Attributs in zahlreichen XSLT-Elementen verwendet.

Nehmen Sie z. B. an, Sie wollen im User-Agent-Header feststellen, welcher Browser benutzt wird. Dafür sollten Sie eine Variable verwenden, die den Wert des User-Agent-Headers enthält:

<xsl:variable name="user-agent" select="fmxslt:get_header('User-Agent')"/>

Für die Erweiterungsfunktionen, die einen Wert zurückgeben, weist der Wert den angegebenen XSLT-Typ auf. Viele Funktionen geben Strings zurück, aber einige Funktionen geben eine Knotengruppe (node-set) zurück, die durchlaufen werden kann.

Hinweis Dieser Abschnitt beschreibt die FileMaker XSLT-Erweiterungsfunktionen und -parameter und bietet einige Beispiele. Zusätzliche Beispiele für jede Funktion finden Sie in der Referenz der FileMaker XSLT-Erweiterungsfunktionen. Weitere Informationen finden Sie unter „Referenz der FileMaker XSLT-Erweiterungsfunktionen auf Seite“ 58.

Erläuterung der FileMaker-spezifischen XSLT-Parameter, die von der Web Publishing Engine gesetzt werden

Beim Verarbeiten einer Abfrage setzt die Web Publishing Engine dynamisch die Werte der folgenden FileMaker-spezifischen XSLT-Parameter. Sie können die Werte dieser Parameter in Ihrem Stylesheet mithilfe des Elements <xsl:param> verwenden.

FileMaker-spezifischer XSLT-Parameter	Weitere Informationen
<xsl:param name="request-query"/>	„Verwenden der Query-Information in einer Abfrage“ im nächsten Abschnitt.
<xsl:param name="client-ip"/> <xsl:param name="client-user-name"/> <xsl:param name="client-password"/>	„Abrufen von Clientinformationen auf Seite“ 67.
<xsl:param name="xml-base-uri"/>	„Verwenden des Basis-URI-Parameters der Web Publishing Engine auf Seite“ 68.
<xsl:param name="authenticated-xml-base-uri">	„Verwenden des Parameters „authenticated base URI“ auf Seite“ 68.

Verwenden der Query-Information in einer Abfrage

Sie können mithilfe eines FileMaker XSLT-Parameters auf Informationen in einer Abfrage in einem URL oder in HTML-Formulardaten zugreifen. Beispielsweise können Sie auf die aktuelle Abfrage-Query-Information zugreifen, um die aktuelle Position in einer Datensatz-Ergebnismenge zu bestimmen und Verknüpfungen zum vorherigen und nächsten Datensatz zu erstellen.

Die folgenden FileMaker XSLT-Parameter bieten Zugriff auf alle Query-Befehle und Query-Parameter, die für Abfragen von FileMaker XML-Daten über die Web Publishing Engine verwendet werden:

```
<xsl:param name="request-query"/>
```

Mit Ausnahme von Feldnamen gibt die Web Publishing Engine alle Namen von Query-Befehlen und Query-Parametern in Kleinbuchstaben zurück. Die Groß-/Kleinschreibung von Feldnamen wird beibehalten.

Ein XML-Dokumentfragment wird in den Abfrage-Query-Parameter in der folgenden Grammatik geladen:

```
<!DOCTYPE query [
    <!ELEMENT query (parameter)*>
        <!ATTLIST query action CDATA #REQUIRED>
        <!ELEMENT parameter (#PCDATA)>
        <!ATTLIST parameter name CDATA #REQUIRED>
        ]
```

Hinweis Die Query-Information befindet sich laut Definition im Namespace fmq="http://www.filemaker.com/xml/query". Stellen Sie sicher, dass Sie eine Deklaration des fmq-Namespace im Element <xsl:stylesheet> am Beginn Ihres XSLT-Stylesheets angeben. Weitere Informationen finden Sie unter „Erläuterung von Namespaces und Präfixen für FileMaker XSLT-Stylesheets auf Seite“ 61.

Nehmen Sie z. B. an, Sie wollen auf die Query-Befehle und Query-Parameter in dieser Abfrage zugreifen:

```
http://192.168.123.101/fmi/xsl/my_stylesheet.xsl?–db=products&–lay=sales&–grammar=fmresultset&–token.1=abc123&–findall
```

Wenn Sie die Anweisung <xsl:param name="request-query" /> vor dem Vorlagenabschnitt („template“) einfügen, speichert die Web Publishing Engine dieses XML-Dokumentfragment in diesem Parameter:

```
<query action="my_stylesheet.xsl" xmlns="http://www.filemaker.com/xml/query">
    <parameter name="–db">products</parameter>
    <parameter name="–lay">sales</parameter>
    <parameter name="–grammar">fmresultset</parameter>
    <parameter name="–token.1">abc123</parameter>
    <parameter name="–findall"></parameter>
</query>
```

Sie können dann den Parameter request-query verwenden, um auf den Wert eines Tokens zuzugreifen, der über einen XPath-Ausdruck in einem URL übergeben wurde. Beispiel:

```
$request-query/fmq:query/fmq:parameter[@name = '–token.1']
```

Abrufen von Clientinformationen

Sie können mithilfe der folgenden FileMaker XSLT-Parameter Informationen über die IP-Adresse, den Benutzernamen und das Passwort eines Web-Clients von der Web Publishing Engine abrufen:

```
<xsl:param name="client-ip"/>
<xsl:param name="client-user-name"/>
<xsl:param name="client-password">
```

Nehmen Sie diese Parameteranweisungen vor dem ersten <xsl:template>-Element in Ihr XSLT-Stylesheet auf.

Diese Parameter identifizieren den Web-Benutzer, wenn ein Stylesheet programmtechnisch weitere passwortgeschützte XML-Dokumente lädt. Weitere Informationen finden Sie unter „Laden zusätzlicher Dokumente auf Seite“ 68. Der Web-Benutzer muss den Benutzernamen und das Passwort anfangs im Dialogfeld „HTTP-Standardauthentifizierung“ angeben. Weitere Informationen finden Sie unter „Voraussetzungen für Web-Benutzer zum Zugriff auf eine geschützte Datenbank auf Seite“ 21.

Weitere Informationen und Beispiele für die Verwendung dieser drei FileMaker XSLT-Parameter finden Sie in der Referenz der FileMaker XSLT-Erweiterungsfunktionen.

Verwenden des Basis-URI-Parameters der Web Publishing Engine

Die Web Publishing Engine definiert den Basis-Uniform Resource Identifier (URI)-Parameter als Host und Port, an dem die Web Publishing Engine installiert ist. Dieser Basis-URI ermöglicht, dass Abfragen von XML-Daten aus FileMaker-Datenbanken in Relation zum Web Publishing Engine-Host aufgelöst werden.

Um auf den Basis-URI der Web Publishing Engine zuzugreifen, nehmen Sie diese Anweisungen vor dem ersten <xsl:template>-Element in Ihr XSLT-Stylesheet auf.

```
<xsl:param name="xml-base-uri"/>
```

Sie können den Basis-URI dann für das aktuelle Stylesheet über die Variable $xml-base-uri immer dann verwenden, wenn Sie eine zusätzliche Abfrage auf FileMaker XML-Daten ausführen müssen. Sie können z. B. den Basis-URI in der folgenden Abfrage von zusätzlichen XML-Daten verwenden:

```
<xsl:variable name="layout_information" select="document(concat($xml-base-uri,'/fmi/xml/FMPXMLLAYOUT.xml?
–db=products&–lay=sales&–view'))" />
```

Verwenden des Parameters „authenticated base URI“

Der Parameter authenticated-xml-base-uri kombiniert die Funktionalität der Parameter client-user-name und client-password mit dem Parameter xml-base-uri:

```
<xsl:param name="authenticated-xml-base-uri"/>
```

Verwenden Sie diesen Parameter, um ein zusätzliches, passwortgeschütztes XML-Dokument zu laden, das den Benutzernamen und das Passwort verlangt, das anfangs in der aktuell verarbeiteten Abfrage eingegeben wurde. Ein Beispiel finden Sie im nächsten Abschnitt, „Laden zusätzlicher Dokumente“.

Nehmen Sie diese Parameteranweisung vor dem ersten <xsl:template>-Element in Ihr XSLT-Stylesheet auf.

Wenn die Werte der Parameter client-user-name und client-password nicht leer sind, lautet der Wert des Parameters authenticated-xml-base-uri:

```
http://username:password@hostname:port
```

Wenn die Werte der Parameter client-user-name und client-password leer sind, ist der Wert des Parameters authenticated-xml-base-uri identisch mit dem Wert des Parameters xml-base-uri.

Laden zusätzlicher Dokumente

Um bei der Verarbeitung eines XML-Stylesheets ein zusätzliches XML-Dokument zu laden, verwenden Sie die XSLT-Standardfunktion document() mit einem URI für das XML-Dokument. Die Funktion document() gibt die angeforderte XML als eine Knotengruppe (node-set) zurück, die in einem <xsl:variable>-Element gespeichert werden kann.

Um ein XML-Dokument zu laden, das Daten aus einer FileMaker-Datenbank enthält, verwenden Sie die Funktion document() mit FileMaker-Query-Befehl und -Parametern. Beispiel:

```
<xsl:variable name="other-data" select="document(concat($xml-base-uri,'/fmi/xml/FMPXMLLAYOUT.xml?
–db=products&–lay=sales&–view'))"/>
```

Um ein zusätzliches, passwortgeschütztes XML-Dokument zu laden, das den Benutzernamen und das Passwort verlangt, das ursprünglich in der aktuell verarbeiteten Abfrage eingegeben wurde, verwenden Sie den Parameter authenticated-xml-base-uri. Dieser Parameter gibt denselben Benutzernamen und dasselbe Passwort als Teil des URI an, der an die Funktion document() übergeben wird.

Beispiel:

```
<xsl:variable name="other-data" select="document(concat($authenticated-xml-base-uri,
'/fmi/xml/FMPXMLLAYOUT.xml?–db=products&–lay=sales&–view'))"/>
```

Um ein passwortgeschütztes XML-Dokument zu öffnen, das einen anderen Benutzernamen und ein anderes Passwort verlangt, als in der übergeordneten Abfrage angegeben wurde, geben Sie anhand der folgenden Syntax den Benutzernamen und das Passwort als Teil des URI an, der an die Funktion document() übergeben wurde:

```
http://username:password@hostname/path?querystring
```

Um ein XML-Dokument zu laden, das nicht auf einer FileMaker-Datenbank basiert, verwenden Sie die Funktion document() ohne FileMaker-Query-Befehle oder Query-Parameter. Beispiel:

```
<xsl:variable name="other-data" select="document('http://server.company.com/data.xml')" />
```

Wenn Sie die Funktion document() mit einem relativen URL verwenden, versucht die Web Publishing Engine, das XML-Dokument von der Position im lokalen Dateisystem zu laden, die dem relativen Speicherort des Stylesheets entspricht. Nehmen Sie z. B. an, dass ein Stylesheet, das sich im Ordner mystylesheets innerhalb des Ordners xslt-template-files befindet, die folgende document()-Funktion mit einem relativen URL enthält:

```
<xsl:variable name="mydoc" select="document('mystylesheets/mydoc.xml')" />
```

Die Web Publishing Engine versucht, mydoc.xml aus dem Ordner mystylesheets im Ordner xslt-template-files im lokalen Dateisystem zu laden.

Hinweis Wenn Sie den Basis-URI der Web Publishing Engine verwenden, um ein Dokument zu laden, unterstützt die Web Publishing Engine nur HTTP. Wenn Sie ein Dokument von einem externen Server laden, unterstützt die Web Publishing Engine sowohl HTTP als auch HTTPS.

Verwenden der Layoutinformation einer Datenbank in einem Stylesheet

Sie können die Layoutinformation einer FileMaker-Datenbank in ein Stylesheet übernehmen, indem Sie die Information mit der FMPXMLLAYOUT-Grammatik abfragen und dann über die XSLT-Funktion document() in eine Variable laden:

```
<xsl:variable name="layout" select="document(concat($xml-base-uri,'/fmi/xml/FMPXMLLAYOUT.xml?–view'))" />
```

Nehmen Sie z. B. an, Sie wollen ein Menü für ein Feld mit dem Namen Color erstellen, das mit den Werten aus einer Werteliste mit dem Namen shirts gefüllt ist, die wiederum in einem Layout in einer FileMaker-Datenbank definiert sind. So können Sie die Layoutinformation mithilfe der Funktion document() in eine XSLT-Variable laden:

```
<xsl:variable name="layout" select="document(concat($xml-base-uri,'/fmi/xml/FMPXMLLAYOUT.xml?–db=products
&–lay=sales&–view'))" />
<select size="1">
    <xsl:attribute name="name">Color</xsl:attribute>
    <option value="">Select One...</option>
```

```
    <xsl:for-each select="$layout/fml:FMPXMLLAYOUT/fml:VALUELISTS/fml:VALUELIST[@NAME = 'shirts']/
    fml:VALUE">
        <option:>
            <xsl:attribute name="value"><xsl:value-of select="."/></xsl:attribute>
            <xsl:value-of select="."/>
        </option>
    </xsl:for-each>
</select>
```

Verwenden von Inhaltspufferung

Wenn Inhaltspufferung deaktiviert ist, streamt die Web Publishing Engine das Ergebnis einer XSLT-Transformation direkt zurück an den Client. Inhaltspufferung ist immer deaktiviert, sofern Sie sie nicht explizit aktivieren. Wenn Sie Inhaltspufferung aktivieren, speichert die Web Publishing Engine den umgewandelten Inhalt, bis die vollständige Transformation beendet ist.

Inhaltspufferung wird für XSLT-Stylesheets benötigt, die Header manipulieren. Da Header vor dem Antworttext geschrieben werden, muss der Text gepuffert werden, damit die hinzugefügte Header-Information eingeschlossen werden kann.

Es gibt vier FileMaker-Erweiterungsfunktionen, für die das XSLT-Umwandlungsergebnis gepuffert werden muss:

- fmxslt:create_session(): Weitere Informationen finden Sie unter „Verwenden der Sitzungserweiterungsfunktionen auf Seite“ 72.
- fmxslt:set_header(): Weitere Informationen finden Sie unter „Verwenden von Header-Funktionen auf Seite“ 76.
- fmxslt:set_status_code (): Weitere Informationen finden Sie unter „Verwenden von Header-Funktionen auf Seite“ 76.
- fmxslt:set_cookie(): Weitere Informationen finden Sie unter „Verwenden der Cookie-Erweiterungsfunktionen auf Seite“ 76.

Damit diese FileMaker-Erweiterungsfunktionen korrekt arbeiten, müssen Sie die folgende XSLT-Verarbeitungsanweisung für die Abfrage in das Dokument der obersten Ebene aufnehmen:

```
<?xslt-cwp-buffer buffer-content="true"?>
```

Wichtig Wenn ein Basis-Stylesheet ein weiteres Stylesheet enthält, muss das Basis-Stylesheet die Verarbeitungsanweisung <?xslt-cwp-buffer?> enthalten. Diese Anweisung wird ignoriert, wenn sie in einem eingeschlossenen Stylesheet verwendet wird.

Ein Vorteil der Verwendung der Verarbeitungsanweisung zur Pufferung der Antwort ist, dass die Web Publishing Engine die Länge der Antwort ermitteln und den Content-Length-Header in der Antwort setzen kann. Durch Puffern der Antwort kann die Leistung der Web Publishing Engine beeinträchtigt werden.

Verwenden von Web Publishing Engine-Sitzungen zum Speichern von Informationen zwischen Abfragen

Mithilfe der Server-seitigen Sitzungen der Web Publishing Engine können Sie jeden Informationstyp zwischen Abfragen verfolgen und speichern. Sitzungen erlauben Ihnen, eine Web-Anwendung zu erstellen, die mithilfe persistenter beliebiger Informationen den Status zwischen Abfragen beibehalten kann. So könnten beispielsweise Benutzer-Client-Informationen, die auf einer ersten Formularseite eingegeben wurden, in einer Sitzung gespeichert und dann zum Ausfüllen der Werte auf einer späteren Seite verwendet werden.

Standardmäßig speichert die Web Publishing Engine die Sitzungs-ID in einem Cookie. Um Clients zu berücksichtigen, die keine Cookies erlauben, können Sie dem URL die Sitzungs-ID mithilfe der Funktion fmxslt:session_encode_url() hinzufügen. Damit in allen Situationen Kompatibilität gewährleistet ist, sollten Sie alle URLs auf der Seite mit der Funktion fmxslt:session_encode_url() kodieren. Diese Funktion fügt Ihrem URL den durch Semikolons getrennten Parameter jsessionid hinzu, der die Kennung für die übergeordnete Sitzung des jeweiligen Clients darstellt.

Anstatt den folgenden Link auf eine Seite zu setzen:

```
<a href="my_stylesheet.xsl?–db=products&–lay=sales&–grammar=fmresultset&–findall">hyperlinked text</a>
```

Sollten Sie alle Links wie folgt auf eine Seite platzieren:

```
<a href="{fmxslt:session_encode_url('my_stylesheet.xsl?–db=products&–lay=sales&–grammar=fmresultset
&–findall')}">hyperlinked text</a>
```

Wenn der Client keine Cookies zulässt, beinhaltet die Seite:

```
<a href="my_stylesheet.xsl;jsessionid=<session id>?–db=products&–lay=sales&–grammar=fmresultset&–findall">
hyperlinked text</a>
```

Wenn die Web Publishing Engine erkennt, dass der Client Cookies erlaubt, speichert die Funktion fmxslt:session_encode_url() die Sitzungs-ID in einem Cookie anstelle eines URLs.

Hinweis Sitzungsinformationen bleiben nach einem Neustart der Web Publishing Engine nicht erhalten.

Verwenden der Sitzungserweiterungsfunktionen

Verwenden Sie die folgenden Erweiterungsfunktionen für Sitzungen, um Sitzungsvariablen zu manipulieren. Sie können einen String, eine Zahl, einen booleschen Wert oder eine Knotengruppe (node-set) in einem Sitzungsobjekt speichern. Mithilfe einer Knotengruppe (node-set) können Sie eine Datenstruktur in XML anlegen und sie dann zwischen Abfragen im Sitzungsobjekt speichern.

Sitzungserweiterungsfunktion	Zurückgegebener Datentyp	Beschreibung
fmxslt:session_exists(String Sitzungsname)	Boolescher Wert	Prüft, ob eine Sitzung mit dem angegebenen Namen existiert.
fmxslt:create_session(String Sitzungsnname)	Boolescher Wert	Erstellt eine Sitzung mit dem angegebenen Sitzungsnamen und der Standard-Zeitüberschreitung, die über die Administrationskonsole gesetzt wird. Weitere Informationen erhalten Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*. **Hinweis** Diese Funktion verlangt die Verarbeitungsanweisung <?xslt-cwp-buffer?>. Weitere Informationen finden Sie unter „Verwenden von Inhaltspufferung auf Seite“ 70.
fmxslt:invalidate_session(Stringsession-name)	Boolescher Wert	Erzwingt eine sofortige Zeitüberschreitung der Sitzung.
fmxslt:set_session_timeout(String Sitzungsname, Number Zeitüberschreitung)	Boolescher Wert	Setzt die Sitzungszeitüberschreitung in Minuten. Die Standardzeitüberschreitung für Sitzungen wird über die Administrationskonsole eingestellt.
fmxslt:session_encode_url(String URL)	String	Kodiert einen URL mit der Sitzungs-ID, wenn der Client keine Cookies unterstützt; gibt andernfalls den eingegebenen URL zurück.
fmxslt:set_session_object(String Sitzungsnname, String Name, Object Wert)	XSLT-Objekt (Zahl, String, Boolescher Wert oder Knotengruppe)	Speichert ein XSLT-Objekt (Zahl, String, Boolescher Wert oder Knotengruppe (node-set)) unter einer Sitzung, die später über die Funktion fmxslt:get_session_object () abgerufen werden kann. Diese Funktion gibt auch das zuvor unter dem angegebenen Sitzungsobjektnamen gespeicherte Objekt zurück. Wenn kein Objekt unter dem Namen gespeichert wurde, wird ein Null-Objekt zurückgegeben.
fmxslt:get_session_object(String Sitzungsname, String Name)	XSLT-Objekt	Bezieht ein XSLT-Objekt aus der Sitzung.
fmxslt:remove_session_object(String Sitzungsname, String Name)	XSLT-Objekt	Gibt ein XSLT-Objekt zurück und entfernt es dann aus der Sitzung.

Beispiel für das Erstellen einer Sitzung und anschließendes Speichern einer bevorzugten Farbe in der Sitzung:

```
<xsl:variable name="session">
    <xsl:choose>
        <xsl:when test="not (fmxslt:session_exists(string($session-name)))">
            <xsl:value-of select="fmxslt:create_session(string($session-name))"/>
        </xsl:when>
        <xsl:otherwise>true</xsl:otherwise>
    </xsl:choose>
</xsl:variable>
<xsl:variable name="favorite-color" select="fmxslt:set_session_object(string($session-name), 'favorite-color', string($color))"/>
```

Wichtig

- Um sicherzustellen, dass Benutzer nach Abschluss einer Sitzung von einer Datenbank abgemeldet werden, erzwingen Sie anhand der Funktion fmxslt:invalidate_session () eine sofortige Zeitüberschreitung der Sitzung.
- Wenn Sie Variablenfelder oder ein Script benutzen, das einen Status festlegt oder ändert, müssen Sie mithilfe der Administrationskonsole die Option XSLT-Datenbanksitzungen für die Web Publishing Engine aktivieren. Andernfalls werden Variablenfeldwerte und Status zwischen Abfragen nicht beibehalten. Weitere Informationen erhalten Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.
- Wenn Sie bei der Verwendung von Web Publishing Engine-Sitzungen in eine andere Datei wechseln, werden globale Feldwerte nicht beibehalten. Die Web Publishing Engine schließt die erste Datei, bevor sie die zweite Datei öffnet. Alternativ können Sie auf Daten aus der zweiten Datenbank zugreifen, indem Sie ein Layout in der ersten Datenbankdatei verwenden.

Senden von E-Mail-Nachrichten von der Web Publishing Engine

Mithilfe der Web Publishing Engine können Sie E-Mail-Nachrichten generieren, was für eigene Web-Lösungen nützlich ist. Damit die Web Publishing Engine eine E-Mail-Nachricht sendet, verwenden Sie eine der folgenden drei fmxslt:send_email ()-Erweiterungsfunktionen in einem XSLT-Stylesheet. Sie können diese Funktionen benutzen, um eine oder mehrere separate Nachrichten zu senden. Da die fmxslt:send_email ()-Funktionen im Server-seitigen XSLT-Stylesheet der Web Publishing Engine enthalten sind, kann ein Client keine unautorisierten E-Mail-Nachrichten über die Web Publishing Engine senden.

E-Mail-Erweiterungsfunktion	Zurückgegebener Datentyp	Beschreibung
fmxslt:send_email(String smtpFelder, String Nachrichtentext)	Boolescher Wert	Sendet eine E-Mail-Nachricht beliebiger Länge in Standardtext aus der Web Publishing Engine und verwendet dafür die Standard-Textkodierung der Web Publishing Engine.
fmxslt:send_email(String smtpFelder, String Nachrichtentext, String Kodierung)	Boolescher Wert	Sendet eine E-Mail-Nachricht beliebiger Länge in Standardtext und verwendet dafür eine der folgenden Textkodierungen: US-ASCII, ISO-8859-1, ISO-8859-15, ISO-2022-JP, Shift_JIS, UTF-8. Informationen über diese Kodierungen finden Sie unter „Festlegen von Textkodierung für Abfragen auf Seite“ 63.
fmxslt:send_email(String smtpFelder, String xsltDatei, Node xml, boolean inklBilder)	Boolescher Wert	Sendet eine HTML-basierte E-Mail-Nachricht mit der Zeichenkodierung, die über das encoding-Attribut des <xsl:output>-Elements im Stylesheet angegeben ist. Wenn das encoding-Attribut nicht im <xsl:output>-Element angegeben wurde, wird für E-Mail-Nachrichten die Standard-Textkodierung der Web Publishing Engine verwendet.

Hinweise

- In jeder der drei Formen der Funktion fmxslt:send_email () ist der Parameter smtpFields ein URL-kodierter String beliebiger Länge mit einer Adresse und Betreffdaten im folgenden Format, das auf RFC 2368, dem mailto-URL-Schema basiert:

  benutzername@host?name1=wert1&name2=wert2...

  Dabei gilt: benutzername@host gibt einen Empfänger an. Die Name-Wert-Paare können in beliebiger Reihenfolge angegeben werden und werden wie folgt definiert:

  - from=benutzername@host (darf nur einmal auftreten). Das from-Feld muss angegeben werden.
  - to=benutzername@host. Verwenden Sie dieses Name-Wert-Paar für zusätzliche Empfänger.
  - reply-to=benutzername@host (kann nur einmal vorhanden sein)
  - cc=benutzername@host
  - bcc=benutzername@host
  - subject=string (kann nur einmal vorhanden sein)

  Wenn die Felder from, reply-to oder subject öfter als einmal angegeben werden, wird die E-Mail-Nachricht nicht gesendet, die Funktion gibt den Wert false() zurück und der entsprechende Fehlerstatus wird gesetzt.

- Die Web Publishing Engine überprüft die Syntax aller angegebenen E-Mail-Adressen. Sie müssen die folgende Form haben:

  user@host.tld oder "genannter Kennzeichner"<user@host.tld>

  Dabei gilt: tld ist eine beliebige Domäne der oberen Ebene wie com oder net. Wenn beliebige dieser Felder eine ungültige E-Mail-Adresse enthalten, wird die E-Mail-Nachricht nicht gesendet und der entsprechende Fehlerstatuscode wird gesetzt.
- Die einzelnen Werte für den Parameter smtpFelder, z. B. der Betreff (subject), müssen URL-kodierte Strings sein. Beispiel: Das Zeichen & muss als & und Leerzeichen als %20 angegeben werden. Der vollständige String für den Parameter smtpFelder muss XML-kodiert sein (siehe das Beispiel am Ende dieses Abschnitts).
- Für jede dieser Funktionen wird der Wert true() zurückgegeben, wenn die E-Mail-Nachricht erfolgreich gesendet wurde; andernfalls wird false() zurückgegeben.
- Für E-Mail-Nachrichten verwendet die Web Publishing Engine anfangs die Textkodierung ISO-8859-1 als Standardeinstellung. Sie können diese Einstellung über die Administrationskonsole ändern. Weitere Informationen erhalten Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.
- Die Funktion fmxslt:send_email(String smtpFelder, String xsltDatei, Node xml, boolean inklBilder) sendet eine E-Mail-Nachricht bestehend aus XML-Daten, die durch das in dieser Funktion angegebene E-Mail-Stylesheet verarbeitet werden.
  - Geben Sie für den Parameter xsltDatei den Namen des E-Mail-Stylesheets an, indem Sie einen URL eingeben, der relativ zur Hauptverarbeitungs-Stylesheet-Datei für die Abfrage ist.
  - Geben Sie für den Parameter xml den übergeordneten Knoten der XML-Daten an, die Sie mit dem E-Mail-Stylesheet verwenden wollen. Um eine E-Mail-Nachricht mit denselben XML-Daten zu senden, die im Browser angezeigt werden, geben Sie einfach den XPath für das Root-Verzeichnis des Dokuments an: „/“. Andernfalls können Sie ein anderes XML-Dokument verwenden, indem Sie es zunächst mithilfe der Funktion document() laden und es dann an die Funktion fmxslt:send_email() übergeben.
  - Geben Sie für den Parameter inklBilder den booleschen Wert true() an, damit die Web Publishing Engine alle Bilder einschließt, die in der HTML der E-Mail-Nachricht als Anlagen angegeben sind. Dieser Parameter schließt sowohl die FileMaker-Datenbankbilder als auch andere Bilder von anderen Speicherorten ein. Die Web Publishing Engine ändert die Bild-URLs so, dass sie auf die Anlagen verweisen. Bei sehr zahlreichen oder großen Bildern kann die Leistung beeinträchtigt werden. Wenn Sie false () angeben, ändert die Web Publishing Engine die URLs für die Bilder nicht. Wenn die URLs absolut sind, versucht der E-Mail-Client, die Bilder vom Web-Server zu laden.

Beispiel für die Verwendung der Funktion fmxslt:send_email(String smtpFelder, String xsltDatei, Node xml, boolean inklBilder) in einer XPath-Anweisung, z. B. im <xsl:if>-Element:

```
fmxslt:send_email('tom_jones@company.com?subject=project%20status&amp;from=john_smith@company.com
&amp;cc=jane_doe@company.com','my_mail_template.xsl', /, true())
```

Informationen über die Konfiguration der Web Publishing Engine für die Verbindung zu einem SMTP-Server finden Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.

Verwenden von Header-Funktionen

Sie können die Funktion fmxslt:get_header() verwenden, um Informationen aus der HTTP-Abfrage und Antwort-Headern zu lesen, und die Funktion fmxslt:set_header(), um Informationen in die Header zu schreiben. Diese Funktionen sind nützlich, wenn der Client mithilfe der Header-Informationen Daten aus dem Web-Server abrufen kann oder wenn Sie aus anderen Gründen einen HTTP-Header einrichten müssen.

Header-Erweiterungsfunktion	Zurückgegebener Datentyp	Beschreibung
fmxslt:get_header(String name)	String	Gibt den angegebenen Header-Wert zurück.
fmxslt:set_header(String name, String wert)	Ungültig	Legt den angegebenen Header-Wert fest.
fmxslt:set_status_code(Number status-code)	Ungültig	Setzt den HTTP-Statuscode.

Hinweise

- Der in den Funktionen fmxslt:get_header() und fmxslt:set_header() verwendete Name sowie der in der Funktion fmxslt:set_header() verwendete Wert können ein String beliebiger Länge sein.
- Die Funktion fmxslt:set_header() und die Funktion fmxslt:set_status_code() verlangen die Verarbeitungsanweisung <?xslt-cwp-buffer?>. Weitere Informationen finden Sie unter „Verwenden von Inhaltspufferung auf Seite“ 70.

Die folgenden Beispiele demonstrieren, wie der Wert des Headers gesetzt wird. Nehmen Sie an, Sie verwenden ein Stylesheet zur Ausgabe einer vCard. Wenn ein Browser versucht, die Stylesheet-Seite zu laden, besteht potenziell das Problem, dass der Browser die .xsl-Datei als Stylesheet anstatt als vCard interpretieren könnte. Wenn Sie den Header „Inhalt-Disposition“ verwenden, können Sie angeben, dass eine Anlage mit der Erweiterung .vcf vorhanden ist.

```
<xsl:value-of select="fmxslt:set_header('Inhalt-Disposition','attachment;filename=test.vcf')" />
```

Verwenden der Cookie-Erweiterungsfunktionen

Mithilfe der Cookie-Erweiterungsfunktionen können Sie Cookies abrufen oder einrichten, die im Webbrowser des Clients gespeichert sind.

Cookie-Erweiterungsfunktion	Zurückgegebener Datentyp	Beschreibung
fmxslt:get_cookie(String name)	Knotengruppe	Gibt die COOKIE-Knotengruppe node-set zurück, die den angegebenen Cookie-Namen enthält.
fmxslt:get_cookies()	Knotengruppe	Gibt die COOKIE-Knotengruppe (node-set) mit allen vom Client gelieferten Cookies zurück.
fmxslt:set_cookie(String name, String wert)	Ungültig	Speichert das angegebene Cookie mit dem angegebenen Wert im Browser des Clients.
fmxslt:set_cookie(String name, String wert, Number ablauf, String pfad, String domäne)	Ungültig	Speichert das angegebene Cookie mit allen für ein Cookie verfügbaren Werten im Browser des Clients. Der Parameter ablauf gibt die Anzahl an Sekunden bis zum Ablauf des Cookies an.

Hinweise

- Die Funktionen fmxslt:get_cookie() und fmxslt:get_cookies() geben eine Knotengruppe (node-set) mit der folgenden Struktur zurück:

```
<!ELEMENT cookies (cookie)*>
    <!ATTLIST cookie xmlns CDATA #FIXED "http://www.filemaker.com/xml/cookie">
<!ELEMENT cookie (#PCDATA)>
    <!ATTLIST cookie name CDATA #REQUIRED>
```

- Der XML-Namespace für die Cookies node-set ist "http://www.filemaker.com/xml/cookie". Stellen Sie sicher, dass Sie den Namespace deklarieren und einen Präfix für den Namespace angeben.
- Alle Parameterwerte für die fmxslt:set_cookie-Funktionen müssen gültig sein, da der Webbrowser sonst die fmxslt:set_cookie-Funktionsabfragen ignoriert.
- Für alle Cookie-Funktionen können die String-Parameter eine beliebige Länge aufweisen.
- Beide Formen der Funktion fmxslt:set_cookie() verlangen die Verarbeitungsanweisung <?xslt-cwp-buffer?>. Weitere Informationen finden Sie unter „Verwenden von Inhaltspufferung auf Seite“ 70.

Beispiel: get_cookie

Das folgende Beispiel ruft einen Cookie preferences und seinen Wert ab:

```
<xsl:variable name="pref_cookie" select="fmxslt:get_cookie('preferences')"/>
<xsl:value-of select="concat('Cookie Name = ', $pref_cookie/fmc:cookies/fmc:cookie/@name)"/> <br/>
<xsl:value-of select="concat('Cookie Value = ', $pref_cookie/fmc:cookies/fmc:cookie)"/>
```

Beispiel: set_cookie

Beispiel für das Setzen eines Cookies mit allen Werten:

```
<xsl:variable name="storing_cookie" select="fmxslt:set_cookie ('text1', 'text2', 1800, 'my_text', 'my.company.com')" />
```

Verwenden der Erweiterungsfunktionen zur Stringmanipulation

Sie können mithilfe der Funktionen zur Stringmanipulation die Kodierung von Strings beliebiger Länge ändern.

Erweiterungsfunktionen zur Stringmanipulation	Zurückgegebener Datentyp	Beschreibung
fmxslt:break_encode(String wert)	String	Gibt einen HTML-Break-kodierten String zurück. Zeichen wie & (kaufmännisches UND) werden durch & ersetzt. Zeilenumbruchzeichen wie Zeilenvorschub und Absatzende-Zeichen werden durch ersetzt. Diese Funktion ist nur wirksam, wenn das Attribut disable-output-escaping der Elemente <xsl:value-of> und <xsl:text> auf „yes“ gesetzt ist (disable-output-escaping="yes"). **Hinweis** Um ein Zeilenvorschub- oder Absatzende-Zeichen in den String einzufügen, auf den die Funktion fmxslt:break_encode() angewendet wird, müssen Sie die folgenden Escape-Zeichen im String verwenden:
(für Zeilenvorschub) bzw.
(für Absatzende). Sie können kein Zeilenvorschub- oder Absatzende-Zeichen in den String einfügen, indem Sie in Ihrem Texteditor den Zeilenschalter drücken.
fmxslt:html_encode(String wert)	String	Gibt einen HTML-kodierten String zurück; Zeichen wie & (kaufmännisches UND) werden durch & ersetzt.
fmxslt:url_encode(String wert)	String	Gibt einen URL-kodierten String zurück. URL-Kodierung wird insbesondere für URLs verwendet, um Zeichen über das Internet zu übertragen. Zum Beispiel ist %26 die URL-kodierte Form von & (kaufmännisches Und). Wenn ein reserviertes Zeichen in Ihrem href verwendet wird, benutzen Sie diese Funktion zur URL-Kodierung Ihres Strings.
fmxslt:url_encode(String wert, String kodierung)	String	Gibt einen URL-kodierten String mit der Zeichenkodierung zurück, die Sie im Parameter kodierung angeben. Mögliche Kodierungen sind: US-ASCII, ISO-8859-1, ISO-8859-15, ISO-2022-JP, Shift_JIS und UTF-8. Verwenden Sie diese Funktion in Situationen, in denen Sie wissen, dass sich die von einem Web-Server erwartete Zeichenkodierung von derjenigen unterscheidet, die in Ihrem aktuellen Stylesheet verwendet wird. Beispiel: Ihre Website-Einstiegsseite kann in UTF-8 angezeigt sein, aber Benutzer klicken auf einen Link, um zu einer japanischen Seite zu wechseln. Wenn die Abfrage japanische Zeichen enthält und die japanischen Seiten Shift_JIS-Kodierung verwenden, wird der String am besten in Shift_JIS kodiert.

Erweiterungsfunktionen zur Stringmanipulation	Zurückgegebener Datentyp	Beschreibung
fmxslt:url_decode(String wert)	String	Gibt einen URL-dekodierten String zurück, der zuvor kodiert war.
fmxslt:url_decode(String wert, String kodierung)	String	Gibt einen URL-dekodierten String mit der Zeichenkodierung zurück, die Sie im Parameter kodierung angeben. Mögliche Kodierungen sind: US-ASCII, ISO-8859-1, ISO-8859-15, ISO-2022-JP, Shift_JIS und UTF-8. Verwenden Sie diese Funktion in Situationen, in denen Sie die in einem URL-kodierten String verwendete Zeichenkodierung angeben müssen, um den String korrekt zu dekodieren. Beispiel: Selbst wenn Ihre Website ISO-8859-1 verwendet, könnten Benutzer ein Formular mit einer anderen Zeichenkodierung senden.

Vergleich von Strings mithilfe regulärer Perl 5-Ausdrücke

Sie können die Erweiterungsfunktion fmxslt:regex_contains() verwenden, um Strings mithilfe regulärer Perl 5-Ausdrücke zu vergleichen. Ein Vergleich mit regulären Ausdrücken ist eine fortschrittliche Art von Textabgleich, bei der Sie bestimmen können, ob ein String einem bestimmten Muster entspricht. Syntax dieser Funktion:

```
fmxslt:regex_contains(String eingabe, String muster)
```

Dabei gilt: eingabe ist ein String und muster ist ein regulärer Perl 5-Ausdruck. Weitere Informationen zur Syntax von regelmäßigen Perl 5-Ausdrücken finden Sie unter www.perldoc.com. Die Funktion fmxslt:regex_contains() gibt einen booleschen Wert zurück.

Diese Funktion ist nützlich, wenn Sie anspruchsvollere Möglichkeiten zur Stringmanipulation benötigen, die von Standard-XSLT nicht zur Verfügung gestellt werden. Sie können z. B. feststellen, ob ein Feldwert eine gültige Telefonnummer oder E-Mail-Adresse enthält, indem Sie den String mit einem regulären Perl 5-Ausdruck vergleichen.

Beispiel für den Einsatz dieser Funktion, um zu ermitteln, ob ein Feldwert korrekt konstruierte E-Mail-Adressen enthält:

```
<xsl:variable name="email" select="'foo@bar.com'"/>
<xsl:if test="fmxslt:regex_contains($email,'^\w+[\w-\.]*\@\w+((-\w+)|(\w*))\.[a-z]{2,3}$')">Valid Email</xsl:if>
```

Wenn die Web Publishing Engine das Muster nicht analysieren kann, wird der Fehlerstatus auf Fehlercode 10311 gesetzt. Weitere Informationen finden Sie unter „Fehlercodenummern für die FileMaker XSLT-Erweiterungsfunktionen auf Seite“ 121.

Prüfen von Werten in einem Feld, das als Kontrollkästchen formatiert ist

Mithilfe der folgenden Erweiterungsfunktion können Sie ermitteln, ob ein bestimmter Wert in einer Kontrollkästchen-Werteliste in einem Feld in der FileMaker-Datenbank gespeichert ist:

fmxslt:contains_checkbox_value(String werteString, String werteListenEintrag)

Dabei gilt: werteString ist ein XPath, der das Feld angibt, und werteListenEintrag ist der Wert, den Sie suchen.

Wenn der angegebene Wert im Feld gespeichert ist, gibt diese boolesche Funktion true() zurück. Andernfalls gibt sie false() zurück. Sie können anhand dieser Funktion ermitteln, ob das geprüfte Attribut in einem HTML-Formular so eingestellt werden soll, dass es ein Kontrollkästchen als markiert zeigt.

Nehmen Sie z. B. an, ein Feld in einem FileMaker-Datenbanklayout hat die folgenden Kontrollkästchenoptionen:

[ ] Rot

[ ] Blau

[ ] Grün

[ ] Klein

[ ] Mittel

[ ] Groß

Wenn ein Benutzer nur „Rot“ auswählt, würde das Feld den String Rot enthalten. Anhand des folgenden Funktionsaufrufs könnten Sie feststellen, ob das Feld Blau enthält:

fmxslt:contains_checkbox_value(<feldwert_knoten>,'Blau')

Dabei gilt: <feldwert_knoten> ist der XPath zum <data>-Element für das Kontrollkästchenfeld. Die Funktion würde in diesem Beispiel false zurückgeben.

Eine übliche Anwendung dieser Funktion ist die Anzeige der Kontrollkästchen-Werteliste in einer Web-Seite und die Auswahl der Kontrollkästchen auf der Web-Seite, die auch in der Datenbank ausgewählt sind. Die folgenden HTML- und XSLT-Anweisungen erstellen eine Gruppe von Kontrollkästchen für ein Feld mit dem Namen style, das eine Werteliste mit dem Namen color_size benutzt:

```
<xsl:variable name="field-value" select="fmrs:field[@name='style']/fmrs:data" />
<xsl:for-each select="$valuelists[@NAME = 'color_size']/fml:VALUE">
        <input type="checkbox">
                <xsl:attribute name="name">style</xsl:attribute>
                <xsl:attribute name="value"><xsl:value-of select="."/></xsl:attribute>
                <xsl:if test="fmxslt:contains_checkbox_value($field-value,.)">
                        <xsl:attribute name="checked">checked</xsl:attribute>
                </xsl:if>
        </input><xsl:value-of select="."/><br/>
</xsl:for-each>
```

Die HTML- und XSLT-Anweisungen im Beispiel würden die folgenden Kontrollkästchen auf einer Web-Seite ausgeben, von denen Rot und Mittel ausgewählt sind:

[x] Rot

[ ] Blau

[ ] Grün

[ ] Klein

[x] Mittel

[ ] Groß

Verwenden der Datums-, Zeit- und Tages-Erweiterungsfunktionen

Mithilfe von Erweiterungsfunktionen können Sie die aktuellen Werte für Datum, Uhrzeit oder Tag abrufen und zwei Datums-, Zeit- oder Tagesangaben vergleichen.

Die Funktionen in der folgenden Tabelle verwenden die „fm“-Formate unabhängig von den regionalen Einstellungen. Die „fm“-Formate sind MM/tt/jjjj für Datum, HH:mm:ss für Zeit und MM/tt/jjjj HH:mm:ss für Zeitstempel.

Um ein anderes oder bevorzugtes Format für die Ausgabewerte zu verwenden, verwenden Sie Berechnungsfunktionen oder JavaScript.

Datums-, Zeit-, Tages-Erweiterungsfunktionen	Zurückgegebener Datentyp	Beschreibung
fmxslt:get_date()	String	Gibt das aktuelle Datum im „fm“-Format zurück.
fmxslt:get_date(String format)	String	Gibt das aktuelle Datum in dem Format zurück, das Sie angeben. Geben Sie den String „short“ oder „long“ für den Parameter format ein.
fmxslt:get_time()	String	Gibt die aktuelle Zeit im „fm“-Format zurück.
fmxslt:get_date(String format)	String	Gibt die aktuelle Uhrzeit in dem Format zurück, das Sie angeben. Geben Sie den String „short“ oder „long“ für den Parameter format ein.
fmxslt:get_day()	String	Gibt den aktuellen Tag im Kurzformat zurück.
fmxslt:get_day(String format)	String	Gibt den aktuellen Tag in dem Format zurück, das Sie angeben. Geben Sie den String „short“ oder „long“ für den Parameter format ein.
fmxslt:get_fm_date_format()	String	Gibt den Formatierungsstring für das „fm“-Datumsformat zurück: „MM/tt/jjjj“
fmxslt:get_short_date_format()	String	Gibt den Formatierungsstring für das kurze Datumsformat zurück: „M/t/jj“
fmxslt:get_long_date_format()	String	Gibt den Formatierungsstring für das lange Datumsformat zurück: „MMM t, jjjj“
fmxslt:get_fm_date_format()	String	Gibt den Formatierungsstring für das „fm“-Zeitformat zurück: „HH:mm:ss “

Datums-, Zeit-, Tages-Erweiterungsfunktionen	Zurückgegebener Datentyp	Beschreibung
fmxslt:get_short_time_format()	String	Gibt den Formatierungsstring für das kurze Zeitformat zurück: „hh:mm“
fmxslt:get_long_time_format()	String	Gibt den Formatierungsstring für das lange Zeitformat zurück: „hh:mm:ss z“
fmxslt:get_short_day_format()	String	Gibt den Formatierungsstring für das kurze Tagesformat zurück: „EEE“
fmxslt:get_long_day_format()	String	Gibt den Formatierungsstring für das lange Tagesformat zurück: „EEEE“
fmxslt:compare_date(String datum1, String datum2)	Zahl	Diese Funktion vergleicht zwei Datumswerte. Sie gibt eine negative Zahl zurück, wenn datum1 vor datum2 liegt. Sie gibt eine positive Zahl zurück, wenn datum1 nach datum2 liegt. Sie gibt eine 0 zurück, wenn datum1 identisch mit datum2 ist. Beide Datumswerte müssen im „fm“-Format angegeben sein.
fmxslt:compare_time(String zeit1, String zeit2)	Zahl	Diese Funktion vergleicht zwei Zeitwerte. Sie gibt eine negative Zahl zurück, wenn zeit1 vor zeit2 liegt. Sie gibt eine positive Zahl zurück, wenn zeit1 nach zeit2 liegt. Sie gibt eine 0 zurück, wenn zeit1 identisch mit zeit2 ist. Beide Zeitwerte müssen im „fm“-Format angegeben sein.
fmxslt:compare_day(String tag1, String tag2)	Zahl	Diese Funktion vergleicht zwei Tageswerte. Sie gibt eine negative Zahl zurück, wenn tag1 vor tag2 liegt. Sie gibt eine positive Zahl zurück, wenn tag1 nach tag2 liegt. Sie gibt eine 0 zurück, wenn tag1 identisch mit tag2 ist. Beide Tage müssen im kurzen Tagesformat angegeben sein.

Die Funktionen in der folgenden Tabelle verwenden spezielle Formatstrings, die ein Datums- und Zeitformat angeben. Weitere Informationen finden Sie im nächsten Abschnitt, „Erläuterung von Datums- und Zeitformatstrings“.

Datums-, Zeit-, Tages-Erweiterungsfunktionen	Zurückgegebener Datentyp	Beschreibung
fmxslt:get_datetime(String datumsFormat)	String	Gibt das aktuelle Datum und die Uhrzeit zurück und verwendet dafür die Datums- und Zeitformatstrings.

Datums-, Zeit-, Tages-Erweiterungsfunktionen	Zurückgegebener Datentyp	Beschreibung
fmxslt:convert_datetime(String altesFormat, String neuesFormat, String datum)	String	Konvertiert das angegebene Datum aus altesFormat in einen String in der Form von neuesFormat. Die Strings altesFormat und neuesFormat müssen mithilfe der Datums- und Zeitformatstrings angegeben werden.
fmxslt:compare_datetime(String datumsFormat1, String datumsFormat2, String datum1, String datum2)	Zahl	Diese Funktion vergleicht datum1 und datum2, indem sie die Datumsangaben gemäß den jeweiligen Datumsformaten dekodiert. Sie gibt eine negative Zahl zurück, wenn datum1 vor datum2 liegt. Sie gibt eine positive Zahl zurück, wenn datum1 nach datum2 liegt. Sie gibt eine 0 zurück, wenn datum1 identisch mit datum2 ist. Die Strings datumsFormat1 und datumsFormat2 müssen mithilfe der Datums- und Zeitformatstrings angegeben werden.

Erläuterung von Datums- und Zeitformatstrings

Die Datums- und Zeitformate werden durch Datums- und Zeitmusterstrings angegeben. Innerhalb von Datums- und Zeitmusterstrings werden Buchstaben ohne Anführungszeichen von A bis Z und von a bis z als Musterbuchstaben interpretiert, die Komponenten des Datums- oder Zeitstrings darstellen.

Die folgenden Musterbuchstaben sind definiert (alle anderen Zeichen von A bis Z und a bis z sind reserviert):

Brief	Datums- oder Zeitkomponente	Darstellung	Beispiele
G	Ärabezeichnung	Text	n. Chr.
y	Jahr	Jahr	1996; 96
M	Monat im Jahr	Monat	Juli; Jul; 07
w	Woche im Jahr	Zahl	27
M	Woche im Monat	Zahl	2
D	Tag im Jahr	Zahl	189
d	Tag im Monat	Zahl	10
F	Tag der Woche im Monat	Zahl	2
E	Wochentag	Text	Mittwoch; Mi
a	AM/PM-Markierung	Text	PM
P	Stunde des Tages (0-23)	Zahl	0
k	Stunde des Tages (1-24)	Zahl	24
K	Stunde vormittags/nachmittags (0-11)	Zahl	0
h	Stunde vormittags/nachmittags (1-12)	Zahl	12
m	Minute in Stunde	Zahl	30
s	Sekunde in Minute	Zahl	55

Brief	Datums- oder Zeitkomponente	Darstellung	Beispiele
S	Millisekunde	Zahl	978
z	Zeitzone	Allgemeine Zeitzone	Pazifische Standardzeit; PST; GMT-08:00
Z	Zeitzone	RFC 822 Zeitzone	-0800

Musterbuchstaben werden gewöhnlich wiederholt, da ihre Anzahl die exakte Darstellung bestimmt:

- **Text**: Wenn die Anzahl der Musterbuchstaben vier oder mehr beträgt, wird für die Formatierung die ausführliche Form verwendet. Andernfalls wird eine kurze oder abgekürzte Form verwendet, falls verfügbar. Beim Parsing werden beide Formen unabhängig von der Anzahl der Musterbuchstaben berücksichtigt.
- **Zahl**: Beim Formatieren gibt die Anzahl der Musterbuchstaben die minimale Stellenanzahl an, Zahlen mit weniger Stellen werden bis zu dieser Anzahl mit Nullen aufgefüllt. Beim Parsing wird die Anzahl der Musterbuchstaben ignoriert, sofern sie nicht zur Trennung zweier benachbarter Felder erforderlich sind.
- **Jahr**: Wenn beim Formatieren zwei Musterbuchstaben vorhanden sind, wird das Jahr auf zwei Stellen abgeschnitten. Andernfalls wird es als eine Zahl interpretiert.

  Wenn beim Parsing mehr als zwei Musterbuchstaben vorhanden sind, wird das Jahr unabhängig von der Stellenanzahl buchstäblich interpretiert. Bei Verwenden des Musters MM/tt/jjjj wird die Angabe 01.11.12 als 11. Jan 12 n. Chr. analysiert.

  Beim Parsing mit dem abgekürzten Jahresmuster (j oder jj) muss das abgekürzte Jahr in Relation zu einem Jahrhundert interpretiert werden, indem Datumsangaben so angepasst werden, dass sie im Zeitraum von 80 Jahren vor und 20 Jahren nach der Erstellung der Instanz liegen. Wenn Sie z. B. das Muster MM/tt/jjjj verwenden und eine Instanz mit diesem Datumsformat am 1. Januar 1997 erstellt wurde, wird der String „01.11.12“ als 11. Januar 2012 interpretiert. Dagegen wird der String „05.04.64“ als 4. Mai 1964 interpretiert. Beim Parsing werden nur Strings mit exakt zwei Stellen als das Standardjahrhundert analysiert. Jeder andere numerische String, z. B. ein einstelliger String, ein drei- oder mehrstelliger String oder ein zweistelliger String, der nicht ausschließlich aus Ziffern besteht (z. B. -1), wird buchstäblich interpretiert. Also werden 01.02.3 oder 01.02.003 mit demselben Muster wie 2. Jan 3 n. Chr. analysiert. Ebenso wird 01.02.-3 als 2. Jan 4 v. Chr. interpretiert.
- **Monat**: Wenn zwei oder mehr Musterbuchstaben vorhanden sind, wird der Monat als Text interpretiert, andernfalls als Zahl.
- **Allgemeine Zeitzone**: Zeitzonen werden als Text interpretiert, wenn sie Namen haben. Für Zeitzonen, die einen von GMT versetzten Wert haben, wird folgende Syntax verwendet:
  - GMTVersatzZeitzone. *GMT Vorzeichen Stunden:Minuten*
  - Vorzeichen. + oder -
  - Stunden. *Ziffer* oder *Ziffer Ziffer*
  - Minuten. *Ziffer Ziffer*
  - Ziffer. Eines von: 0 1 2 3 4 5 6 7 8 9

  Stunden müssen zwischen 0 und 23 liegen, *Minuten* zwischen 00 und 59. Das Format ist unabhängig von der Region und die Ziffern müssen dem Basic Latin-Block des Unicode-Standards entnommen werden.

Beim Parsing werden auch RFC 822-Zeitzonen akzeptiert.

- **RFC 822-Zeitzone**: Beim Formatieren wird das vierstellige RFC 822-Zeitzonenformat verwendet:
  - RFC822Zeitzone. *Vorzeichen ZweiStelligeStunden Minuten*
  - ZweiStelligeStunden. *Ziffer Ziffer*

  ZweiStelligeStunden muss zwischen 00 und 23 liegen. Andere Definitionen sind wie für allgemeine Zeitzonen.

  Beim Parsing werden auch allgemeine Zeitzonen akzeptiert.

Die folgenden Beispiele zeigen, wie Datums- und Zeitmuster in den USA interpretiert werden. Die angegebenen Datums- und Zeitwerte sind 2001-07-04 12:08:56 Ortszeit in der US-amerikanischen Zeitzone „Pacific Time“.

Datums- und Zeitmuster	Ergebnis
"yyyy.MM.dd G 'at' HH:mm:ss z"	2001.07.04 n. Chr. at 12:08:56 PDT
"EEE, MMM d, ' 'yy"	Mi, Jul 4, '01
"h:mm a"	12:08 PM
"hh 'o' 'clock' a, zzzz"	12 o'clock PM, Pazifische Sommerzeit
"K:mm a, z"	0:08 PM, PDT
"yyyyy.MMMMM.dd GGG hh:mm aaa"	02001.Juli.04 n. Chr. 12:08 PM
"EEE, d MMM yyyy HH:mm:ss Z"	Mi, 4 Jul 2001 12:08:56 -0700
"yyMMddHHmmssZ"	010704120856-0700

Copyright 2003 Sun Microsystems, Inc. Genehmigter Nachdruck.

Prüfen des Fehlerstatus von Erweiterungsfunktionen

Sie können mithilfe der folgenden Erweiterungsfunktion in einem XSLT-Stylesheet den aktuellen Fehlerstatus der zuletzt aufgerufenen FileMaker XSLT-Erweiterungsfunktion prüfen und Fehler behandeln, die bei der Verarbeitung Ihrer Seiten auftreten:

```
fmxslt:check_error_status()
```

Beim Aufruf der Funktion fmxslt:check_error_status() gibt die Web Publishing Engine den aktuellen Fehlercode für die zuletzt aufgerufene Funktion im Typ Zahl zurück und setzt den Fehlerstatus dann auf 0 („Kein Fehler“) zurück. Informationen über die Fehlercodewerte finden Sie unter „Fehlercodenummern für die FileMaker XSLT-Erweiterungsfunktionen auf Seite“ 121.

Verwenden der Protokollierung

Sie können mithilfe des XSLT-Standardelements <xsl:message> Protokolleinträge in die Protokolldatei der Web Publishing Engine schreiben. Weitere Informationen finden Sie unter „Verwenden des Web Publishing Engine-Programmprotokolls auf Seite“ 93.

Verwenden der Server-seitigen Verarbeitung von Scriptingsprachen

Der zugrunde liegende XSLT-Transformer, der in die Web Publishing Engine integriert ist, unterstützt die Server-seitige Verarbeitung von Scriptingsprachen. Daher können Sie JavaScript verwenden, um Ihre eigenen Erweiterungsfunktionen zu entwickeln, die direkt aus einem XSLT-Stylesheet aufgerufen werden können.

Mit FileMaker Server 8 Advanced werden zwei Java-Libraries installiert, um diese Funktionalität zu ermöglichen:

- bsf.jar – Diese Library ermöglicht dem XSLT-Transformer, Verbindungen zu Scriptingsprachen herzustellen.
- js.jsr – Diese Library ist eine vollständige JavaScript-Implementierung des Mozilla-Projekts.

Mit diesen Libraries können Sie Ihre eigenen Erweiterungsfunktionen innerhalb Ihres XSLT-Stylesheet-Codes erstellen. Diese Erweiterungsfunktionen können jede Scriptinglogik implementieren und sind besser zu verwalten, als XSLT und XPath zu verwenden, um logische Funktionen umzusetzen.

Weitere Informationen über die Erweiterungsunterstützung des XSLT-Transformers finden Sie auf der Apache Xalan Extensions-Website:

http://xml.apache.org/xalan-j/extensions.html

Definieren einer Erweiterungsfunktion

So definieren Sie eine Erweiterungsfunktion innerhalb Ihres Stylesheets:

1. Definieren Sie den Namespace für die Erweiterung.

   Fügen Sie den Namespace xalan hinzu, um den XSLT-Transformer anzuweisen, Erweiterungskomponenten zu unterstützen, indem Sie den Namen Ihres eigenen Erweiterungsfunktion-Namespace angeben. Das folgende Beispiel verwendet fmp-ex als Namespace-Präfix der Erweiterungsfunktion.

   ```
   <xsl:stylesheet version="1.0"
   xmlns:xsl=http://www.w3.org/1999/XSL/Transform
   xmlns:xalan=http://xml.apache.org/xslt
   xmlns:fmp-ex="ext1"
   exclude-result-prefixes="xsl xalan fmp-ex">
   ```

2. Definieren Sie die Erweiterungskomponente und die Erweiterungsfunktionen mit dem Code, der tatsächlich Ihre Erweiterungsfunktion implementiert.

   ```
   <xalan:component prefix="fmp-ex" functions="getValueColor">
   <xalan:script lang="javascript">
         function getValueColor(value) {
         if (value > 0)
               return ("#009900");
         else
               return ("#CC0000");
         }
   </xalan:script>
   </xalan:component>
   ```

Dieses Beispiel gibt einen Farbwert basierend auf einem Eingabewert zurück. Wenn der Eingabewert größer als 0 ist, ist die zurückgegebene Farbe Grün ("#009900"). Wenn andernfalls der Wert kleiner 0 ist, ist die zurückgegebene Farbe Rot ("#CC0000").

Hinweis Das Element <xalan:component> muss dem Element <xsl:stylesheet> untergeordnet sein.

3. Verwenden Sie die Erweiterungsfunktion innerhalb des Stylesheets.

   Das folgende Beispiel zeigt, wie Sie eine Erweiterungsfunktion mittels einer Xpath-Anweisung aufrufen.

   Das erste Beispiel würde die Schriftfarbe auf Grün ("#009900") setzen.

   ```
   <font color="{fmp-ex:getValueColor(50)}">Der Wert ist 50</font>
   ```

   Das zweite Beispiel würde die Schriftfarbe auf Rot ("##CC0000") setzen.

   ```
   <font color="{fmp-ex:getValueColor(-500)}">Der Wert ist -500</font>
   ```

Ein Beispiel für eine Erweiterungsfunktion

Die einfache JavaScript-Funktion, die im obigen Prozess verwendet wird, könnte über eine <xsl:choose>-Anweisung implementiert worden sein. Aber der Kniff bei der Verwendung einer Scriptingerweiterung liegt darin, dass Sie eine Funktion erstellen können, die nicht alleine in XSLT oder XPath implementiert werden kann.

Nehmen wir zum Beispiel an, dass Sie ein Intranet-Portal für Ihr Unternehmen aufsetzen und Sie die aktuellen Aktiennotierungen auf dieser Portalseite aufnehmen möchten. Es gibt zwar XML-Börsen-Feeds, im Allgemeinen sind jedoch kommerzielle Lizenzen erforderlich, um auf sie zuzugreifen. Sie können jedoch Börsendaten in einem CSV-Dokument von Yahoo herunterladen. Die XPath document()-Funktion kann Inhalt aus XML-Quellen importieren, aber Sie müssten den CSV-Inhalt zu XML konvertieren. Eine Lösung besteht darin, dass Sie JavaScript verwenden, um die CSV-Börsendaten herunterzuladen, die Datei zu parsen und die Daten zu extrahieren.

Dieser URL zeigt die Syntax für den Abruf einer Börsennotierung aus Yahoo in Form einer CSV-Datei:

```
http://quote.yahoo.com/d/quotes.csv?s=<ticker>&f=l1gh&e=.csv
```

Dabei gilt: <ticker> steht für das Börsensymbol, für das Sie Daten abrufen möchten.

Die zurückgegebenen Daten bestehen aus drei Zahlen, die durch Kommata getrennt sind. Beispiel:

```
31.79,31.17,32.12
```

Der erste Wert ist der letzte Handelspreis, der zweite Wert das Tagestief und der dritte Wert das Tageshoch.

Das folgende Beispiel zeigt eine JavaScript-XSLT-Erweiterungsfunktion, die einen aktuellen Börsenkurs von Yahoo abruft und ihn über eine XPath-Funktion zur Verfügung stellt:

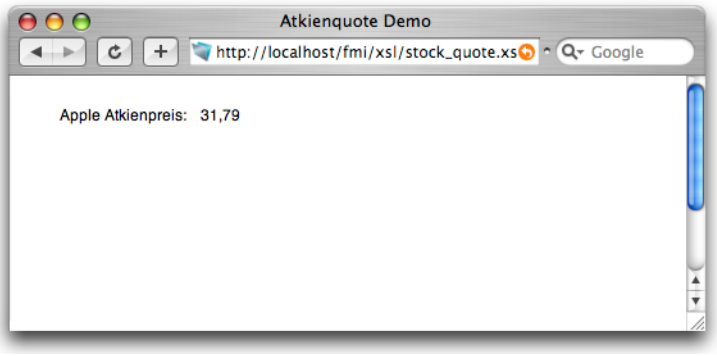
```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
	exclude-result-prefixes="xsl fmxslt fmrs xalan fmp-ex"
	version="1.0"
	xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
	xmlns:fmrs="http://www.filemaker.com/xml/fmresultset"
	xmlns:fmxslt="xalan://com.fmi.xslt.ExtensionFunctions"
	xmlns:xalan="http://xml.apache.org/xslt"
	xmlns:fmp-ex="ext1"
>

<?xslt-cwp-query params="-grammar=fmresultset&-process" ?>
<xsl:output method="html"/>
<xalan:component prefix="fmp-ex" functions="include get_quote" >
<xalan:script lang="javascript">
	function include(url) {
		var dest = new java.net.URL(url);
		var dis = new java.io.DataInputStream(dest.openStream());
		var res = "";
		while ((line = dis.readLine()) != null)
		{
			res += line + java.lang.System.getProperty("line.separator");
		}
		dis.close();
		return res;
	}
	function get_quote(ticker) {
		url = "http://quote.yahoo.com/d/quotes.csv?s=”+
			"+ticker+"&amp;amp;f=l1gh&amp;amp;e=.csv";
		csv_file = include(url);
		var str_tokenizer = new java.util.StringTokenizer(csv_file, ',');
		// Der erste Token ist der letzte Handelspreis.
		var last = str_tokenizer.nextToken();
		return last;
	}
</xalan:script>
</xalan:component>
```

```
<xsl:template match="/fmrs:fmresultset">
    <html>
        <body>
        <font size="2" face="verdana, arial">
        Apple Stock Price: <xsl:value-of select="fmp-ex:get_quote('AAPL')"/>
        </font>
        </body>
    </html>
</xsl:template>

</xsl:stylesheet>
```

Wenn die Web Publishing Engine dieses Stylesheet verarbeitet, ruft es die Börsendaten von Yahoo ab. Die Funktion get_quote() geht durch die Börsenkursdaten und gibt die Daten an das Stylesheet zurück. Die transformierte Ausgabe wird im Browser angezeigt:



Kapitel 6
Testen und Überwachen einer Site

Testen Sie Ihre Custom Web Publishing-Site, bevor Sie sie einsetzen. Sie können die Site mithilfe von Protokolldateien während der Tests und nach der Freigabe überwachen.

Testen einer Custom Web Publishing-Site

Bevor Sie die Benutzer benachrichtigen, dass Ihre Custom Web Publishing-Site verfügbar ist, müssen Sie unbedingt sicherstellen, dass sie wie gewünscht aussieht und funktioniert.

- Testen Sie Funktionen wie das Suchen, Hinzufügen, Löschen und Sortieren von Datensätzen mit unterschiedlichen Konten und Zugriffsrechten.
- Überprüfen Sie, ob sich die verschiedenen Zugriffsrechte wie erwartet verhalten, indem Sie sich mit verschiedenen Konten anmelden. Stellen Sie sicher, dass unbefugte Benutzer nicht auf Ihre Daten zugreifen oder sie verändern können.
- Überprüfen Sie alle Scripts, um sicherzustellen, dass das Ergebnis wie erwartet ist (siehe „FileMaker-Scripts und Custom Web Publishing“ auf Seite 17 für Informationen zur Gestaltung Web-freundlicher Scripts).
- Testen Sie Ihre Site mit unterschiedlichen Betriebssystemen und Webbrowsern.

Hinweis Wenn Sie über keine Netzwerkverbindung verfügen und den Web-Server, die Web Publishing Engine und FileMaker Server auf einem einzigen Computer installiert haben, können Sie Ihre Custom Web Publishing-Site testen, indem Sie im URL Folgendes verwenden: http://localhost/ oder http://127.0.0.1/ Information über die URL-Syntax finden Sie unter „Erläuterung der URL-Syntax für XML-Daten und Medienobjekte“ auf Seite 28, und „Erläuterung der URL-Syntax für FileMaker XSLT-Stylesheets“ auf Seite 58.

Beispiele für Stylesheets zum Testen der XML-Ausgabe

Hier sind zwei Beispiele für XSLT-Stylesheets, die sich zum Testen von XML-Ausgabe eignen.

- Das folgende Stylesheet-Beispiel gibt die angeforderten XML-Daten ohne Umwandlung aus. Dieses Stylesheet ist nützlich für die Anzeige der tatsächlichen XML-Daten, die die Web Publishing Engine verwendet.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fmrs="http://www.filemaker.com/xml/fmresultset">
    <xsl:output method="xml"/>
    <xsl:template match="/">
        <xsl:copy-of select="."/>
    </xsl:template>
</xsl:stylesheet>
```

- Bei der Fehlerbehebung in einem Stylesheet können Sie das folgende Beispiel des HTML-Tags <textarea> verwenden, um das XML-Quelldokument anzuzeigen, auf das über das Stylesheet in einem abrollbaren Textbereich zugegriffen wurde. Auf derselben Seite können Sie die umgewandelten XSLT-Ergebnisse mit dem XML-Quelldokument vor der Umwandlung vergleichen.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fmrs="http://www.filemaker.com/xml/fmresultset">
    <xsl:output method="html"/>
<html>
    <Datenbereich>
    <xsl:template match="/fmrs:fmresultset">
        <textarea rows="20" cols="100">
            <xsl:copy-of select="."/>
        </textarea><br/>
    </xsl:template>
    </body>
</html>
</xsl:stylesheet>
```

Überwachen Ihrer Website

Sie können anhand der folgenden Typen von Protokolldateien Ihre Custom Web Publishing-Site überwachen und Informationen über Web-Benutzer sammeln, die Ihre Site besuchen:

- Web-Server-Zugriff und Fehlerprotokolle
- Web Publishing Engine-Programmprotokoll
- Web-Server-Modul-Fehlerprotokoll
- interne Web Publishing Core-Zugriffsprotokolle

Verwenden der Web-Server-Zugriffs- und Fehlerprotokolle

Der Apache-Web-Server generiert eine Zugriffsprotokolldatei und eine Fehlerprotokolldatei. Die Apache-Zugriffsprotokolldatei, standardmäßig im W3C Common Logfile Format, ist eine Aufzeichnung aller beim Web-Server eingehenden HTTP-Abfragen. Das Apache-Fehlerprotokoll ist eine Aufzeichnung von Problemen bei der Verarbeitung von HTTP-Abfragen. Weitere Informationen über diese Protokolldateien finden Sie in der Dokumentation zu Ihrem Apache-Web-Server.

Der Microsoft IIS-Web-Server generiert eine Zugriffsprotokolldatei und zeigt Fehler in der Windows-Ereignisanzeige, anstatt sie in eine Protokolldatei zu schreiben. Die Zugriffsprotokolldatei, standardmäßig im W3C Extended Log File Format, ist eine Aufzeichnung aller beim Web-Server eingehenden HTTP-Abfragen. Sie können auch das W3C Common Logfile Format für das Zugriffsprotokoll verwenden. Weitere Informationen finden Sie in der Dokumentation zum Microsoft IIS-Web-Server.

Informationen über das W3C Common Logfile Format und das W3C Extended Log File Format finden Sie auf der Website des World Wide Web Consortium: www.w3.org.

Verwenden des Web Publishing Engine-Programmprotokolls

Standardmäßig generiert die Web Publishing Engine eine Programmprotokolldatei, die eine Aufzeichnung von Web Publishing Engine-Fehler-, Script- und Benutzerprotokollinformationen darstellt.

- Die Fehlerprotokollinformation beschreibt alle ungewöhnlichen Web Publishing Engine-Fehler, die aufgetreten sind. Dem Web-Benutzer häufig gemeldete Fehler wie „Datenbank nicht geöffnet“ werden nicht aufgezeichnet.
- Die Scriptprotokollinformation beschreibt alle Fehler, die generiert wurden, während Web-Benutzer Scripts ausführten. Sie listet z. B. Scriptschritte auf, die übersprungen wurden, weil sie nicht Web-kompatibel sind.
- Die Benutzerprotokollmeldungen umfassen Meldungen, die vom XSLT-Element <xsl:message> in XSLT-Stylesheets erzeugt wurden. Immer wenn Web-Benutzer auf Ihr XSLT-Stylesheet zugreifen, werden in der Programmprotokolldatei Informationen aufgezeichnet, die Sie in einem <xsl:message>-Element angegeben haben Weitere Informationen finden Sie unter Kapitel 5, „Entwickeln von FileMaker XSLT-Stylesheets“.

Das Programmprotokoll heißt pe_application_log.txt und befindet sich im Ordner Logs, der sich im Web Publishing-Ordner im FileMaker Server-Ordner auf dem Web Publishing Engine-Host befindet.

Die Datei pe_application_log.txt wird generiert, wenn beliebige der folgenden Protokollierungsoptionen in der Web Publishing Engine aktiviert sind:

- Fehlerprotokollierung
- Scriptprotokollierung
- Benutzerprotokollierung

Alle drei Protokollierungsoptionen sind standardmäßig aktiviert. Weitere Informationen über das Einstellen dieser Optionen über die Administrationskonsole finden Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.

Hinweis Die Einträge im Programmprotokoll werden nicht automatisch gelöscht, daher kann die Datei mit der Zeit sehr umfangreich werden. Um Festplattenspeicherplatz auf Ihrem Host-Computer zu sparen, empfiehlt sich ein regelmäßiges Archivieren der Programmprotokolldatei.

Verwenden des Web-Server-Modul-Fehlerprotokolls

Wenn der Web-Server keine Verbindung zur Web Publishing Engine aufbauen kann, generiert das Web Server Modul eine Protokolldatei, die eine Aufzeichnung aller Fehler bei ihrer Operation darstellt. Diese Datei heißt web_server_module_log.txt und befindet sich im Ordner Logs, der im Web Publishing-Ordner im FileMaker Server-Ordner auf dem Web-Server-Host gespeichert ist.

Verwenden von internen Web Publishing Core-Zugriffsprotokollen

Die Softwarekomponente Web Publishing Core der Web Publishing Engine generiert zwei interne Zugriffsprotokolldateien, die jeden Zugriff auf den Web Publishing Core aufzeichnen:

- Das Zugriffsprotokoll wpc_access_log.txt enthält eine Aufzeichnung aller Endanwenderabfragen zur Generierung von XML und Verwendung von FileMaker Server Instant Web Publishing. Diese Abfragen werden vom Web-Server direkt an den Web Publishing Core geleitet.
- Das Zugriffsprotokoll pe_internal_access_log.txt enthält eine Aufzeichnung aller internen XML-Abfragen, die die Softwarekomponente XSLT-CWP der Web Publishing Engine ausführt, während XSLT-Abfragen verarbeitet werden. Diese Abfragen werden intern in der Web Publishing Engine von der Softwarekomponente XSLT-CWP an die Softwarekomponente Web Publishing Core geleitet.

Diese Protokolldateien befinden sich im Ordner Logs, der im Web Publishing-Ordner im FileMaker Server-Ordner auf dem Web Publishing Engine-Host gespeichert ist.

Die internen Zugriffsprotokolle werden generiert, wenn die Option Anmeldungsprotokollierung in der Web Publishing Engine aktiviert ist. Die Standardeinstellung ist aktiviert. Informationen über das Einstellen der Option Anmeldungsprotokollierung über die Administrationskonsole finden Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.

Anhang A
Gültige Namen in Query-Strings

Dieser Anhang beschreibt die gültigen Namen für Query-Befehle und deren Parameter, die Sie in einem Query-String verwenden können, wenn Sie über die Web Publishing Engine auf FileMaker-Daten zugreifen.

Erläuterung von Query-Befehlen und -Parametern

Die folgende Liste enthält sämtliche Namen für Query-Befehle und Query-Parameter:

Query-Befehlsnamen	Query-Parameternamen
–dbnames (Siehe Seite 100.)	–db (Siehe Seite 103.)
–delete (Siehe Seite 100.)	–encoding (nur XSLT) (Siehe Seite 103.)
–dup (Siehe Seite 100.)	–field (Siehe Seite 104.)
–edit (Siehe Seite 101.)	fieldname (Siehe Seite 104.)
–find, –findall, –findany (Siehe Seite 101.)	fieldname.op (Siehe Seite 105.)
–layoutnames (Siehe Seite 102.)	–grammar (nur XSLT) (Siehe Seite 106.)
–new (Siehe Seite 102.)	–lay (Siehe Seite 106.)
–process (nur XSLT) (Siehe Seite 102.)	–lay.response (Siehe Seite 106.)
–scriptnames (Siehe Seite 102.)	–lop (Siehe Seite 107.)
–view (Siehe Seite 103.)	–max (Siehe Seite 107.)
	–modid (Siehe Seite 107.)
	–recid (Siehe Seite 107.)
	–script (Siehe Seite 108.)
	–script.param (Siehe Seite 108.)
	–script.prefind (Siehe Seite 108.)
	–script.prefind.param (Siehe Seite 109.)
	–script.presort (Siehe Seite 109.)
	–script.presort.param (Siehe Seite 109.)
	–skip (Siehe Seite 110.)
	–sortfield.[1-9] (Siehe Seite 110.)
	–sortorder.[1-9] (Siehe Seite 110.)
	–stylehref (Siehe Seite 111.)
	–styletype (Siehe Seite 112.)
	–token.[string] (nur XSLT) (Siehe Seite 112.)

Wichtig Der Parameter –lay für die Angabe eines Datenbanklayouts ist bei allen Query-Befehlen erforderlich mit Ausnahme von –dbnames, –layoutnames, –scriptnames und –process (nur XSLT-Abfragen).

Veraltete Abfragenamen und Parameter

Die folgenden Abfragenamen und Parameter werden nicht mehr unterstützt:

Veraltete Abfragenamen	Kommentar
-dbopen	Öffnete eine Datenbank im Web-Ordner, wobei die Option „Entfernte Verwaltung“ aktiviert war.
-dbclose	Schließt eine Datenbank im Web-Ordner, wobei die Option „Entfernte Verwaltung“ aktiviert war.
–img	Rief das angegebene Bild ab. Verwenden Sie stattdessen einen Medien-URL Weitere Informationen hierzu finden Sie unter „Erläuterung der URL-Syntax für FileMaker-Medienobjekte in XML-Lösungen“ auf Seite 30, und „Erläuterung der URL-Syntax für FileMaker-Medienobjekte in XSLT-Lösungen“ auf Seite 59.

Veraltete Parameternamen	Kommentar
–password	Gab das Passwort der Datenbank für –dbopen an. Die Web Publishing Engine verwendet HTTP-Standardauthentifizierung Weitere Informationen finden Sie unter „Voraussetzungen für Web-Benutzer zum Zugriff auf eine geschützte Datenbank“ auf Seite 21.
–format	Gab eine Formatdatei für CDML oder XML-Grammatik an. CDML wird nicht mehr unterstützt Weitere Informationen finden Sie unter Anhang C, „Konvertieren von CDML-Lösungen in FileMaker XSLT“.
–fmtfield	Bezog Formatdateiinformationen aus einem Feld in der Datenbank.
–error –errnum –errfmtfield	Für CDML-Fehlerbehandlung verwendet. Die Web Publishing Engine behandelt Fehler im Stylesheet Weitere Informationen hierzu finden Sie unter Anhang B, „Fehlercodes für Custom Web Publishing“, und „Konvertierung der CDML-Variablen-Tags –error und –errornum“ auf Seite 129.
–mailto –mailfrom –mailhost –mailformat –mailcc –mailbcc –mailsub	Verwendet für E-Mail-Nachrichten vom Web Companion. Verwenden Sie FileMaker XSLT-Erweiterungsfunktionen für E-Mail Weitere Informationen finden Sie unter „Senden von E-Mail-Nachrichten von der Web Publishing Engine“ auf Seite 74.

Richtlinien für die Verwendung von Query-Befehlen und -Parametern

Beachten Sie folgende Richtlinien, wenn Sie Query-Befehle und -Parameter in einem Query-String verwenden:

- Ein Query-String darf nur einen einzigen Query-Befehl enthalten. Ein Query-String kann z. B. –new enthalten, um einen neuen Datensatz hinzuzufügen, aber derselbe Query-String kann nicht gleichzeitig –new und –edit enthalten.
- Die meisten Query-Befehle verlangen mehrere entsprechende Query-Parameter im Query-String. Beispielsweise verlangen alle Query-Befehle außer –dbnames und –process den Parameter –db, der die abzufragende Datenbank angibt. (Siehe dazu die Tabellen der erforderlichen Parameter in „Abfragen von XML-Daten mithilfe von FileMaker-Query-Strings“ auf Seite 40.)

- Geben Sie für Query-Parameter und Feldnamen den jeweils gewünschten Wert an, z. B. –db=employees. Geben Sie für Query-Befehle kein =-Zeichen oder einen Wert nach dem Befehlsnamen wie –findall an.
- Query-Befehls- und -Parameternamen müssen in Kleinbuchstaben, z. B. –delete oder –lay, angegeben werden.
- Datenbanknamen, Layoutnamen und Feldnamen, die in Query-Strings verwendet werden, unterscheiden keine Groß- und Kleinschreibung, z. B. gibt –lay=meinlayout den Layoutnamen MeinLayout an.

  Hinweis Feld- und Datenbanknamen, die in XSLT-Anweisungen außerhalb von Query-Strings verwendet werden, unterscheiden Groß-/Kleinschreibung und müssen exakt mit den tatsächlich in der Datenbank verwendeten Namen übereinstimmen. Z. B. in dieser Anweisung:

  ```
  <xsl:value-of select="fmrs:field[@name='LastName']"/>
  ```

  Der Feldverweis LastName in diesem Beispiel muss exakt mit dem Namen des Felds LastName in der Datenbank übereinstimmen.

- Auf Feldnamen, die einen Punkt enthalten (z. B. text.feld), kann nicht über XML oder XSLT mittels einer HTTP-Abfrage zugegriffen werden. Der Punkt ist ein reserviertes Zeichen, das für Datensatz-IDs wie in Abschnitt „Erläuterung der Syntax für einen voll qualifizierten Feldnamen“ nachfolgend beschrieben verwendet wird.
- Für den Befehl –find wird die Groß-/Kleinschreibung eines Feldwerts nicht berücksichtigt. Sie können beispielsweise Field1=Blue oder Field1=blue verwenden. Für die Befehle –new und –edit wird die verwendete Schreibung für einen Feldwert beibehalten und exakt so in der Datenbank gespeichert wie im Query-String angegeben. Beispiel: LastName=Doe.

Erläuterung der Referenz von FileMaker-Query-Strings

Die FileMaker Server Web Publishing-CD enthält eine FileMaker-Datenbank mit dem Namen Query Strings Reference.fp7, die kurze Beschreibungen und Beispiele für jeden der FileMaker-Query-Befehle und -Query-Parameter bietet. Sie können diese Referenz der FileMaker-Query-Strings im Ordner Custom Web Publishing Reference auf der FileMaker Server Web Publishing-CD öffnen.

Erläuterung der Syntax für einen voll qualifizierten Feldnamen

Ein vollständig qualifizierter Feldname identifiziert eine exakte Instanz eines Felds. Da Felder mit gleichen Namen auf unterschiedlichen Tabellen basieren können, müssen Sie vollständig qualifizierte Namen verwenden, um Fehler zu vermeiden.

Syntax für einen vollständig qualifizierten Feldnamen:

```
tabellen-name::feld-name(wiederholung-nummer).datensatz-id
```

Dabei gilt:

- tabellen-name ist der Name der Tabelle, die das Feld enthält. Der Tabellenname ist nur erforderlich, wenn sich das Feld nicht in der zugrunde liegenden Tabelle des im Query-String angegebenen Layouts befindet.
- feld-name(wiederholung-nummer) ist der spezifische Wert in einem Wiederholfeld und ist nur für Wiederholfelder erforderlich. Die Wiederholungsnummer wird ab der Ziffer 1 gezählt. Beispielsweise gibt feld-name(2) den zweiten Wert in einem Wiederholfeld an. Wenn Sie keine Wiederholungsnummer für ein Wiederholfeld angeben, wird der erste Wert des Wiederholfelds verwendet. Die Wiederholungsnummer ist für die Query-Befehle –new und –edit erforderlich, wenn sie Wiederholfelder verwenden, jedoch nicht für den Befehl –find.
- datensatz-id gibt die Datensatz-ID an. Diese ist nur erforderlich, wenn Sie einen Query-String verwenden, um Datensätze in einem Ausschnitt hinzuzufügen oder zu bearbeiten. Weitere Informationen finden Sie in den Abschnitten „Hinzufügen von Datensätzen in einem Ausschnitt“, und „Bearbeiten von Datensätzen in einem Ausschnitt“. Die record-id ist für die Query-Befehle –new und –edit erforderlich, wenn sie Ausschnittfelder verwenden, jedoch nicht für den Befehl –find.

Hinweis Damit sie verfügbar sind, müssen Felder auf das Layout gesetzt werden, das Sie im Query-String angeben.

Hinzufügen von Datensätzen in einem Ausschnitt

Um einem Ausschnitt gleichzeitig mit dem Hinzufügen eines Ausgangsdatensatzes einen neuen Datensatz hinzuzufügen, verwenden Sie den Query-Befehl –new und führen den folgenden Query-String für die Abfrage aus:

- Verwenden Sie den vollständig qualifizierten Feldnamen für das Bezugsfeld im Ausschnitt.
- Geben Sie 0 als Datensatz-ID nach dem Namen des Bezugsfelds im Ausschnitt an.
- Geben Sie mindestens eines der Felder für den Ausgangsdatensatz an, bevor Sie das Bezugsfeld im Ausschnitt angeben.
- Geben Sie die Daten für das Abgleichsfeld (Schlüsselfeld) im Ausgangsdatensatz an.

Beispiel: Der folgende URL fügt einen neuen Employee-Ausgangsdatensatz für John Doe und gleichzeitig einen neuen Bezugsdatensatz für Jane im Ausschnitt hinzu. Der Name der Bezugstabelle ist „Dependents“ und der Name des Bezugsfelds im Ausschnitt lautet „Names“. Das Abgleichsfeld ID speichert eine Angestelltenkennung.

```
http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=family&FirstName=John&LastName=Doe
&ID=9756&Dependents::Names.0=Jane&–new
```

Hinweis Sie können pro Abfrage nur einen Bezugsdatensatz hinzufügen.

Bearbeiten von Datensätzen in einem Ausschnitt

Zur Bearbeitung eines oder mehrerer Datensätze in einem Ausschnitt verwenden Sie den Befehl –edit und eine Datensatz-ID, um den Ausgangsdatensatz anzugeben, der die zu bearbeitenden Ausschnitt-Datensätze enthält. Geben Sie den entsprechenden zu bearbeitenden Ausschnitt-Datensatz anhand seiner Datensatz-ID in einem vollständig qualifizierten Feldnamen an. Sie können eine Datensatz-ID dem Datensatz-ID-Attribut des <record>-Elements im <relatedset>-Element in den XML-Daten entnehmen. Weitere Informationen finden Sie unter „Verwenden der fmresultset-Grammatik“ auf Seite 33.

Beispiel: Der folgende URL bearbeitet einen Datensatz in einem Ausschnitt, wobei der Ausgangsdatensatz die Datensatz-ID 1001 hat. Der Name der Bezugstabelle ist „Dependents“, der Name des Bezugsfelds im Ausschnitt lautet „Names“ und die 2 in „Names.2“ ist die Datensatz-ID des Ausschnitt-Datensatzes.

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=family&–recid=1001
&Dependents::Names.2=Kevin&–edit

Das folgende Beispiel zeigt, wie Sie eine Abfrage verwenden können, um mehrere Ausschnitt-Datensätze über den Ausgangsdatensatz zu bearbeiten:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=family&–recid=1001
&Dependents::Names.2=Kevin&Dependents::Names.5=Susan&–edit

Sie können auch den Befehl –edit verwenden und 0 als ID des Ausschnitt-Datensatzes angeben, um dem Ausschnitt einen neuen Bezugsdatensatz für einen bestehenden Ausgangsdatensatz hinzuzufügen. Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=family&–recid=1001
&Dependents::Names.0=Timothy&–edit

Erläuterung der Syntax für das Angeben eines Variablenfelds

Syntax für das Angeben eines Variablenfelds:

tabellen-name::feld-name(wiederholung-nummer).global

Dabei gilt: global gibt ein Feld an, das global gespeichert wird. Informationen über tabellen-name und feld-name(wiederholung-nummer) finden Sie unter „Erläuterung der Syntax für einen voll qualifizierten Feldnamen“ auf Seite 97. Informationen über Variablenfelder finden Sie in der FileMaker Pro Hilfe.

Sie müssen die .global-Syntax verwenden, um ein Variablenfeld in einem Query-String zu identifizieren. Die Web Publishing Engine setzt die Parameterwerte für Variablenfelder, bevor sie den Query-Befehl ausführt oder andere Parameterwerte im Query-String setzt. Für direkte XML-Abfragen sowie Abfragen, die über XSLT-Stylesheets erfolgen und keine Sitzungen verwenden, laufen die globalen Werte sofort nach Ausführung der Abfrage ab. Für Abfragen, die über XSLT-Stylesheets erfolgen und Sitzungen verwenden, bleiben die globalen Werte für die Dauer der Sitzung erhalten, die im Stylesheet definiert ist, oder bis sie erneut durch eine andere Abfrage geändert werden.

Wenn Sie ein Variablenfeld in einem Query-String nicht mit der .global-Syntax identifizieren, berechnet die Web Publishing Engine das Variablenfeld zusammen mit dem restlichen Query-String, ohne zuvor den Wert für das Variablenfeld zu setzen.

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&Country.global=USA&–edit

Hinweis Wenn Sie den Wert eines Variablenfelds mithilfe des Befehls –edit setzen, erübrigt sich der Parameter –recid, sofern Sie mit dieser Abfrage nur den Wert des Variablenfelds setzen.

Wichtig Wenn Sie ein Variablenfeld in einem XSLT-Stylesheet benutzen, müssen Sie mithilfe der Administrationskonsole die Option XSLT-Datenbanksitzungen für die Web Publishing Engine aktivieren. Andernfalls werden Variablenfeldwerte zwischen Abfragen nicht beibehalten. Weitere Informationen finden Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.

Verwenden der Query-Befehle

Dieser Abschnitt enthält Informationen über die Query-Befehle, die für XML- und XSLT-Abfragen zur Verfügung stehen.

Hinweis Nur in XSLT-Abfragen erfordern alle folgenden Query-Befehle den Query-Parameter –grammar.

Query-Befehl –dbnames (Datenbanknamen)

Ruft die Namen aller Datenbanken ab, die FileMaker Server bereitstellt und die für Custom Web Publishing mit XML oder XSLT aktiviert sind.

Erforderliche Query-Parameter: (keine)

Beispiel:

So rufen Sie die Datenbanknamen ab:

http://192.168.123.101/fmi/xml/fmresultset.xml?–dbnames

Query-Befehl –delete (Datensatz löschen)

Löscht den vom –recid-Parameter angegebenen Datensatz.

Erforderliche Query-Parameter: –db, –lay, –recid

Optionaler Query-Parameter: –script

Beispiel:

So löschen Sie einen Datensatz:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–recid=4&–delete

Query-Befehl –dup (Datensatz duplizieren)

Dupliziert den durch –recid angegebenen Datensatz.

Erforderliche Query-Parameter: –db, –lay, –recid

Optionaler Query-Parameter: –script

Beispiel:

So duplizieren Sie den angegebenen Datensatz:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–recid=14&–dup

Query-Befehl –edit (Datensatz bearbeiten)

Aktualisiert den vom Parameter –recid angegebenen Datensatz und füllt die Felder mit dem Inhalt etwaiger Feldname-Wert-Paare. Der –recid-Parameter gibt an, welcher Datensatz bearbeitet werden sollte.

Erforderliche Query-Parameter: –db, –lay, –recid, ein oder mehrere Feldname(n)

Optionaler Query-Parameter: –modid, –script

Hinweis Informationen über das Bearbeiten von Datensätzen in einem Ausschnitt finden Sie unter „Bearbeiten von Datensätzen in einem Ausschnitt“ auf Seite 99.

Beispiel:

So bearbeiten Sie einen Datensatz:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–recid=13&Country=USA&–edit

Query-Befehle –find, –findall oder –findany (Datensätze suchen)

Sendet eine Suchabfrage mit definierten Kriterien.

Erforderliche Query-Parameter: –db, –lay

Optionale Query-Parameter: –recid, –lop, –op, –max, –skip, –sortorder, –sortfield, –script, –script.prefind, –script.presort, Feldname

Beispiele:

So suchen Sie einen Datensatz nach Feldname:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=family&Country=USA&–find

Hinweis Die mehrfache Angabe eines Feldnamens in einer einzelnen Abfrage wird nicht unterstützt. FileMaker Server geht alle Werte durch, verwendet aber nur den zuletzt geparsten Wert.

So suchen Sie einen Datensatz nach Datensatz-ID:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=family&–recid=427&–find

Um alle Datensätze in der Datenbank zu finden, verwenden Sie –findall:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=family&–findall

Um einen zufällig ermittelten Datensatz zu finden, verwenden Sie –findany:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=family&–findany

–layoutnames (Layoutnamen) (Query-Befehl)

Ruft die Namen aller verfügbaren Layouts für eine angegebene Datenbank ab, die FileMaker Server bereitstellt und die für Custom Web Publishing mit XML oder XSLT aktiviert sind.

Erforderliche Query-Parameter: –db

Beispiel:

So rufen Sie die Namen verfügbarer Layouts ab:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–layoutnames

–new (Neuer Datensatz) (Query-Befehl)

Erzeugt einen neuen Datensatz und fügt in diesen den Inhalt jeglicher Feldnamen/Wert-Paare ein.

Erforderliche Query-Parameter: –db, –lay

Optionaler Query-Parameter: ein oder mehrere Feldname(n), –script

Hinweis Informationen über das Einfügen neuer Daten für einen Ausschnitt finden Sie unter „Hinzufügen von Datensätzen in einem Ausschnitt“ auf Seite 98.

Beispiel:

So fügen Sie einen neuen Datensatz hinzu:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&Country=Australia&–new

-process (XSLT-Stylesheets verarbeiten)

Verarbeitet ein XSLT-Stylesheet, ohne Daten von der Datenbank anzufordern. Dieser Query-Befehl kann nur mit XSLT-Stylesheets verwendet werden.

Erforderlicher Query-Parameter: –grammar Sie müssen die fmresultset- oder FMPXMLRESULT-Grammatik verwenden.

Beispiel:

http://192.168.123.101/fmi/xsl/my_template/my_stylesheet.xsl?–grammar=fmresultset&–process

Weitere Informationen finden Sie unter „Verarbeitung von XSLT-Abfragen, die nicht FileMaker Server abfragen“ auf Seite 64.

–scriptnames (Scriptnamen) (Query-Befehl)

Ruft die Namen aller verfügbaren Scripts für eine angegebene Datenbank ab, die FileMaker Server bereitstellt und die für Custom Web Publishing mit XML oder XSLT aktiviert sind.

Erforderliche Query-Parameter: –db

Beispiel:

So rufen Sie die Namen aller Scripts ab:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–scriptnames

Query-Befehl –view (Layoutinformationen anzeigen)

Wenn die FMPXMLLAYOUT-Grammatik angegeben ist: Ruft Layoutinformationen aus einer Datenbank ab und zeigt sie in der FMPXMLLAYOUT-Grammatik an. Wenn eine Datengrammatik (fmresultset oder FMPXMLRESULT) angegeben ist: Ruft den Metadaten-Abschnitt des XML-Dokuments und einen leeren „recordset“ ab.

Erforderliche Query-Parameter: –db, –lay

Optionaler Query-Parameter: –script

Beispiele:

So rufen Sie Layoutinformationen ab:

http://192.168.123.101/fmi/xml/FMPXMLLAYOUT.xml?–db=employees&–lay=departments&–view

So rufen Sie Metadaten-Informationen ab:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–view

Verwenden der Query-Parameter

Dieser Abschnitt enthält Informationen über die Query-Parameter, die für XML- und XSLT-Abfragen zur Verfügung stehen. Informationen über Parameter, die nur für XSLT-Abfragen verfügbar sind, finden Sie unter „Verwenden von Query-Strings in FileMaker XSLT-Stylesheets“ auf Seite 60.

Query-Parameter –db (Datenbankname)

Gibt die Datenbank an, auf die der Query-Befehl angewendet wird.

Wert: Name der Datenbank ohne etwaige Dateinamenerweiterung.

Hinweis Geben Sie mit dem Namen der Datenbank für den Parameter –db keine Dateinamenerweiterung an. Der tatsächliche Datenbank-Dateiname kann optional eine Erweiterung aufweisen, aber Erweiterungen sind als Wert für den Parameter –db nicht erlaubt.

Erforderlich für: Alle Query-Befehle mit Ausnahme von –dbnames und –process

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–findall

Query-Parameter –encoding (XSLT-Abfrage-Kodierung)

Gibt die Textkodierung für eine XSLT-Abfrage an. Dieser Query-Befehl kann nur für Custom Web Publishing mit XSLT-Abfragen verwendet werden.

Wert: US-ASCII, ISO-8859-1, ISO-8859-15, ISO-2022-JP, Shift_JIS, or UTF-8

Optional für: Alle Query-Befehle in einer XSLT-Abfrage

Beispiel:

http://192.168.123.101/fmi/xsl/my_template/my_stylesheet.xsl?–db=employees&–lay=departments
&–grammar=fmresultset&–encoding=Shift_JIS&–findall

Weitere Informationen finden Sie unter „Festlegen von Textkodierung für Abfragen“ auf Seite 63.

Query-Parameter –field (Medienfeldname)

Gibt den Namen eines Medienfelds zurück.

Erforderlich für: Abfrage von Daten in einem Medienfeld.

Weitere Informationen hierzu finden Sie unter „Erläuterung der URL-Syntax für FileMaker-Medienobjekte in XML-Lösungen“ auf Seite 30, und „Erläuterung der URL-Syntax für FileMaker-Medienobjekte in XSLT-Lösungen“ auf Seite 59.

Query-Parameter „fieldname“ (Nicht-Medienfeldname)

Feldnamen dienen als Kontrollkriterien für den Query-Befehl –find oder zum Ändern des Inhalts eines Datensatzes. Wenn ein Query-Befehl oder -Parameter einen Wert für ein Nicht-Medienfeld verlangt, verwenden Sie den Feldnamen ohne den Bindestrich (–) als den Namensteil des Name-Wert-Paars.

Name: Name des Felds in der FileMaker-Datenbank. Wenn das Feld sich nicht in der zugrunde liegenden Tabelle des Layouts befindet, das im Query-String angegeben ist, muss der Feldname vollständig qualifiziert sein. Auf Feldnamen, die einen Punkt enthalten (z. B. text.feld), kann nicht über XML oder XSLT mittels einer HTTP-Abfrage zugegriffen werden. Der Punkt ist ein reserviertes Zeichen, das für Datensatz-IDs in einem vollständig qualifizierten Feldnamen verwendet wird. Weitere Informationen finden Sie unter „Erläuterung der Syntax für einen voll qualifizierten Feldnamen“ auf Seite 97.

Wert: Geben Sie für die Query-Befehle –new und –edit den Wert an, den Sie im aktuellen Datensatz in dem Feld speichern wollen. Geben Sie für den Query-Befehl –find den Wert an, den Sie im Feld suchen wollen. Wenn Sie den Wert für ein Datums-, Zeit- oder Zeitstempelfeld angeben, müssen Sie den Wert gemäß „fm“-Format für den Feldtyp angeben. Die „fm“-Formate sind MM/tt/jjjj für Datum, HH:mm:ss für Zeit und MM/tt/jjjj HH:mm:ss für Zeitstempel.

Erforderlich für: –edit (Query-Befehl)

Optional für: –new- und –find-Query-Befehle

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–op=eq&FirstName=Sam&–max=1&–find

Hinweis Die mehrfache Angabe eines Feldnamens in einer einzelnen Abfrage wird nicht unterstützt. FileMaker Server geht alle Werte durch, verwendet aber nur den zuletzt geparsten Wert.

Query-Parameter fieldname.op (Vergleichsoperator)

Gibt den Vergleichsoperator an, der auf den Feldnamen, der dem Operator vorangeht, angewendet werden soll. Vergleichsoperatoren werden mit dem Query-Befehl –find verwendet.

Wert: Der zu verwendende Operator. Der Standard-Operator lautet „begins with“. Gültige Operatoren sind:

Schlüsselwort	Äquivalenter Operator in FileMaker Pro
eq	=Wort
cn	*Wort*
bw	Wort*
ew	*Wort
gt	> Wort
gte	>= Wort
lt	< Wort
lte	<= Wort
neq	ausschließen, Wort

Optional für: –find (Query-Befehl)

Erfordert: Einen Feldnamen und einen Wert

Syntax für das Angeben eines Vergleichsoperators:

tabellen-name::feld-name=wert&tabelle-name::feld-name.op=op-symbol

Dabei gilt:

- tabelle-name ist die Tabelle, die das Feld enthält, und ist nur erforderlich, wenn das Feld nicht in der Quelltabelle des im Query-String angegebenen Layouts ist.
- op-symbol ist eines der Schlüsselwörter in der vorherigen Tabelle, wie z. B. cn.

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&name=Tim&name.op=cn&–find

Sie können jeden FileMaker Pro-Suchoperator verwenden, indem Sie das Schlüsselwort bw angeben. Um z. B. den Wertebereich über den Bereichsoperator (...) zu ändern, geben Sie das Schlüsselwort bw an und setzen dann die Zeichen „...“ vor das Suchkriterium.

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&IDnum=915...925&IDnum.op=bw&–find

Weitere Informationen über Operatoren für die Suche nach Text finden Sie in der FileMaker Pro Hilfe.

Query-Parameter –grammar (Grammatik für XSLT-Stylesheets)

Gibt die Grammatik für ein XSLT-Stylesheet an. Dieser Query-Befehl kann nur für Custom Web Publishing mit XSLT-Abfragen verwendet werden.

Wert: fmresultset, FMPXMLRESULT oder FMPXMLLAYOUT

Erforderlich für: Alle XSLT-Abfragen

Beispiel:

http://192.168.123.101/fmi/xsl/my_template/my_stylesheet.xsl?–grammar=fmresultset&–db=mydatabase&–lay=mylayout&–findall

Weitere Informationen finden Sie unter „Angeben einer XML-Grammatik für ein FileMaker XSLT-Stylesheet“ auf Seite 60.

Query-Parameter –lay (Layout)

Gibt das Datenbanklayout an, das Sie verwenden wollen.

Wert: Name des Layouts.

Erforderlich für: Alle Query-Befehle mit Ausnahme von –dbnames, –layoutnames, –scriptnames und –process (nur XSLT-Abfragen)

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–view

Query-Parameter –lay.response (Layout wechseln für Antwort)

Gibt an, dass FileMaker Server beim Verarbeiten einer Abfrage das Layout verwenden soll, das der Parameter –lay angibt, und bei der Verarbeitung der XML-Antwort in das Layout wechseln soll, das der Parameter –lay.response angibt.

Wenn Sie den Parameter –lay.response nicht angeben, verwendet FileMaker Server das vom Parameter –lay angegebene Layout für die Verarbeitung der Abfrage und der Antwort.

Sie können den Parameter –lay.response für XML-Abfragen oder in XSLT-Stylesheet-Abfragen verwenden.

Wert: Name des Layouts.

Optional für: Alle Query-Befehle mit Ausnahme von –dbnames, –layoutnames, –scriptnames und –process (nur XSLT-Abfragen)

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=Budget&Salary=100000&Salary.op=gt&–find&–lay.response=ExecList

Query-Parameter –lop (Logischer Operator)

Gibt an, wie die Suchkriterien im Query-Befehl –find als and- oder or-Suche kombiniert werden.

Wert: and oder or (müssen in Kleinbuchstaben angegeben werden). Wenn der Query-Parameter –lop nicht angegeben wird, verwendet der Query-Befehl –find den Wert and.

Optional für: –find (Query-Befehl)

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&Last+Name=Smith&Birthdate=2/5/1972&–lop=and&–find

Query-Parameter –max (Maximum Datensätze)

Gibt die maximale Anzahl an Datensätzen an, die zurückgegeben werden soll.

Wert: Eine Zahl oder – für die Rückgabe aller Datensätze – der Wert all. Der Wert all muss in Kleinbuchstaben angegeben werden. Wenn –max nicht angegeben wird, werden alle Datensätze zurückgegeben.

Optional für: –find- oder –findall-Query-Befehle

Beispiele:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–max=10&–findall

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–max=all&–findall

Query-Parameter –modid (Änderungs-ID)

Die Änderungs-ID ist ein inkrementeller Zähler, der die aktuelle Version eines Datensatzes angibt. Indem Sie mit dem Query-Befehl –edit eine Änderungs-ID angeben, können Sie sicherstellen, dass Sie die aktuelle Version eines Datensatzes bearbeiten. Wenn der angegebene Wert der Änderungs-ID nicht dem aktuellen Wert der Änderungs-ID in der Datenbank entspricht, wird der Query-Befehl –edit nicht zugelassen und ein Fehlercode wird zurückgegeben.

Wert: Eine Änderungs-ID, die die aktuelle Version eines Datensatzes in einer FileMaker-Datenbank eindeutig identifiziert.

Optional bei: –edit (Query-Befehl)

Erfordert: –recid-Parameter

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–recid=22&–modid=6&last_name=Jones&–edit

Query-Parameter –recid (Datensatz-ID)

Gibt den Datensatz an, der verarbeitet werden soll. Wird hauptsächlich mit den Query-Befehlen –edit und –delete verwendet.

Wert: Eine Datensatz-ID, die einen Datensatz in einer FileMaker-Datenbank eindeutig angibt.

Erforderlich für: –edit-, –delete- und –dup-Query-Befehle

Optional bei: –find (Query-Befehl)

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–recid=22&–delete

–script (Script) (Query-Parameter)

Gibt das FileMaker-Script an, das nach Ausführung des Query-Befehls und der Sortierung ablaufen soll. Weitere Informationen finden Sie unter „Erläuterung der Verarbeitung einer XML-Abfrage“ auf Seite 43.

Wert: Scriptname.

Optional für: Alle Query-Befehle mit Ausnahme von –dbnames, –layoutnames, –scriptnames und –process

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–script=myscript&–findall

Query-Parameter –script.param (Parameter an Script übergeben)

Übergibt einen Parameter an das durch –script angegebene FileMaker-Script.

Wert: Ein einzelner Textparameter.

- Um mehrere Parameter zu übergeben, können Sie einen String erstellen, der die Parameter begrenzt, und Ihr Script die einzelnen Parameter auslesen lassen. Übergeben Sie z. B. param1|param2|param3 als Liste mit dem Zeichen |, das so kodiert ist: param1%7Cparam2%7Cparam3
- Um den Textparameter als Wert zu behandeln, der kein Text ist, kann Ihr Script den Textwert konvertieren. Um z. B. den Textwert in eine Zahl zu konvertieren, könnte Ihr Script Folgendes beinhalten: LiesAlsZahl (Hole(ScriptParam))
- Wenn Ihre Query –script.param ohne –script enthält, wird –script.param ignoriert.
- Wenn Ihre Query mehr als einen –script.param enthält, verwendet die Web Publishing Engine den letzten geparsten Wert.

Optional für: –script

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–script=myscript
&–script.param=Smith%7CChatterjee%7CSu&–findall

Query-Parameter –script.prefind (Script vor Suchen)

Gibt das FileMaker-Script an, das vor dem Suchen und Sortieren von Datensätzen (falls angegeben) während der Verarbeitung des Query-Befehls –find ausgeführt wird.

Wert: Scriptname.

Optional für: Alle Query-Befehle mit Ausnahme von –dbnames, –layoutnames, –scriptnames und –process

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–script.prefind=myscript&–findall

Query-Parameter –script.prefind.param (Parameter an Script vor Suchen übergeben)

Übergibt einen Parameter an das durch –script.prefind angegebene FileMaker-Script.

Wert: Ein einzelner Textparameter.

- Um mehrere Parameter zu übergeben, können Sie einen String erstellen, der die Parameter begrenzt, und Ihr Script die einzelnen Parameter auslesen lassen. Übergeben Sie z. B. param1|param2|param3 als Liste mit dem Zeichen |, das so kodiert ist: param1%7Cparam2%7Cparam3
- Um den Textparameter als Wert zu behandeln, der kein Text ist, kann Ihr Script den Textwert konvertieren. Um z. B. den Textwert in eine Zahl zu konvertieren, könnte Ihr Script Folgendes beinhalten: LiesAlsZahl (Hole(ScriptParam))
- Wenn Ihre Query –script.prefind.param ohne –script.prefind enthält, wird –script.prefind.param ignoriert.
- Wenn Ihre Query mehr als einen –script.prefind.param enthält, verwendet die Web Publishing Engine den letzten geparsten Wert.

Optional für: –script.prefind

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–script.prefind=myscript&–script.prefind.param=payroll&–findall

Query-Parameter –script.presort (Script vor Sortieren)

Gibt das FileMaker-Script an, das nach dem Suchen von Datensätzen (falls angegeben) und vor dem Sortieren der Datensätze während der Verarbeitung des Query-Befehls –find ausgeführt wird.

Optional für: Alle Query-Befehle mit Ausnahme von –dbnames, –layoutnames, –scriptnames und –process

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–script.presort=myscript&–sortfield.1=dept&–sortfield.2=rating&–findall

Query-Parameter –script.presort.param (Parameter an Script vor Sortieren übergeben)

Übergibt einen Parameter an das durch –script.presort angegebene FileMaker-Script.

Wert: Ein einzelner Textparameter.

- Um mehrere Parameter zu übergeben, können Sie einen String erstellen, der die Parameter begrenzt, und Ihr Script die einzelnen Parameter auslesen lassen. Übergeben Sie z. B. param1|param2|param3 als Liste mit dem Zeichen |, das so kodiert ist: param1%7Cparam2%7Cparam3
- Um den Textparameter als Wert zu behandeln, der kein Text ist, kann Ihr Script den Textwert konvertieren. Um z. B. den Textwert in eine Zahl zu konvertieren, könnte Ihr Script Folgendes beinhalten: LiesAlsZahl (Hole(ScriptParam))
- Wenn Ihre Query –script.presort.param ohne –script.presort enthält, wird –script.presort.param ignoriert.
- Wenn Ihre Query mehr als einen –script.presort.param enthält, verwendet die Web Publishing Engine den letzten geparsten Wert.

Optional für: –script.presort

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–script.presort=myscript&–script.presort.param=18%7C65&–sortfield.1=dept&–sortfield.2=rating&–findall

Query-Parameter –skip (Datensätze überspringen)

Gibt an, wie viele Datensätze in der Ergebnismenge übersprungen werden sollen.

Wert: Eine Zahl. Wenn der Wert größer als die Anzahl der Datensätze in der Ergebnismenge ist, wird kein Datensatz angezeigt. Der Standardwert beträgt 0.

Optional für: –find (Query-Befehl)

Im folgenden Beispiel werden die ersten 10 Datensätze in der Ergebnismenge ausgelassen und die Datensätze 11 bis 15 ausgegeben.

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–skip=10&–max=5&–findall

Query-Parameter –sortfield (Sortierfeld)

Gibt das Feld an, das für das Sortieren verwendet wird.

Wert: Feldname

Optional für: –find- oder –findall-Query-Befehle

Der Query-Parameter –sortfield kann mehrfach verwendet werden, um mehrere Feldsortierungen vorzunehmen. Syntax für die Rangfolge der Sortierfelder:

–sortfield.precedence-number=fully-qualified-field-name

Dabei gilt: rang-nummer im Query-Parameter –sortfield.rang-nummer ist eine Zahl (beginnend mit 1), die die Rangfolge für mehrere Sortierfelder angibt.

In folgendem Beispiel wird zuerst das Feld dept sortiert und dann das Feld rating. Beide Felder werden aufsteigend sortiert, da der Query-Parameter –sortorder nicht angegeben wurde.

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=performance&–sortfield.1=dept
&–sortfield.2=rating&–findall

Query-Parameter –sortorder (Sortierfolge)

Gibt die Richtung einer Sortierung an.

Wert: Die Sortierfolge. Gültige Sortierfolgen (wobei <wertelisten-name> der Name einer Werteliste wie Spezial ist):

Schlüsselwort	Äquivalenter Operator in FileMaker Pro
ascend	Sortieren von a bis z, -10 bis 10
descend	Sortieren von z bis a, 10 bis -10
<wertelisten-name>	Sortieren nach der angegebenen Werteliste für das Feld im Layout

Optional für: –find- oder –findall-Query-Befehle

Erfordert: –sortfield (Query-Parameter)

Der Query-Parameter –sortorder kann mit dem Query-Parameter –sortfield verwendet werden, um die Sortierfolge mehrerer Sortierfelder anzugeben. Syntax für die Sortierfolge eines Sortierfelds:

–sortorder.precedence-number=sort-method

Dabei gilt:

- precedence-number im Query-Parameter –sortorder.precedence-number ist eine Zahl von 1 bis 9, die den Query-Parameter –sortfield angibt, auf den sich der Query-Parameter –sortorder bezieht.
- sortier-methode ist eines der Schlüsselwörter in obiger Tabelle für die Festlegung der Sortierfolge, z. B. „ascend“.

Im folgenden Beispiel ist die Sortierfolge für das Sortierfeld mit der höchsten Rangnummer (dept) „ascend“ und für das Sortierfeld mit der zweithöchsten Rangnummer (rating) „descend“. Die precedence-number 2 in –sortorder.2 gibt an, dass der Query-Parameter –sortorder.2=descend sich auf den Query-Parameter –sortfield.2=rating bezieht.

Beispiel:

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=performance&–sortfield.1=dept &–sortorder.1=ascend&–sortfield.2=rating&–sortorder.2=descend&–findall

Hinweis Ein Sortierfeld, für das der Query-Parameter –sortorder nicht angegeben ist, verwendet standardmäßig aufsteigende Sortierung.

Query-Parameter –stylehref (Stil-Href)

Generiert eine Verarbeitungsanweisung für XML-Stylesheets im Ausgabedokument. Dabei setzt er den Wert des href-Attributs (href=/mystylesheet.css or href=/stylesheets/mystylesheet.xsl), damit Sie Client-seitige Cascading Stylesheets (CSS) oder XSLT-Stylesheets mit Ihrem XML-Dokument verwenden können. Der Wert des Parameters –stylehref muss ein absoluter Pfad sein. Das Stylesheet kann einen beliebigen Namen haben, aber die Erweiterung muss .css oder .xsl sein. Weitere Informationen finden Sie unter „Verwenden von Server-seitiger und Client-seitiger Stylesheet-Verarbeitung“ auf Seite 43. Dieser Parameter wird in Verbindung mit dem –styletype-Parameter verwendet.

Optional für: Alle Query-Befehle

Erfordert: –styletype-Parameter

Beispiel (setzt voraus, dass sich mystylesheet.xsl im Root-Ordner der Web-Server-Software befindet):

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–styletype=text/xsl &–stylehref=/mystylesheet.xsl&–findall

Query-Parameter –styletype (Stilart)

Generiert eine Verarbeitungsanweisung für XML-Stylesheets im Ausgabedokument. Dabei setzt er den Wert des Typattributs (type=text/css oder type=text/xsl), damit Sie Cascading Stylesheets (CSS) oder XSLT-Dokumente mit Ihrem XML-Dokument verwenden können. Weitere Informationen finden Sie unter „Verwenden von Server-seitiger und Client-seitiger Stylesheet-Verarbeitung“ auf Seite 43.
Dieser Parameter wird in Verbindung mit dem –stylehref-Parameter verwendet.

Optional für: Alle Query-Befehle

Erfordert: –stylehref-Parameter

Beispiel (setzt voraus, dass sich mystylesheet.css im Root-Ordner der Web-Server-Software befindet):

http://192.168.123.101/fmi/xml/fmresultset.xml?–db=employees&–lay=departments&–styletype=text/css&–stylehref=/mystylesheet.css&–findall

Query-Parameter –token.[string] (Werte zwischen XSLT-Stylesheets übergeben)

Übergibt beliebige benutzerdefinierte Informationen zwischen XSLT-Stylesheets, ohne Sitzungen oder Cookies zu verwenden. Dieser Query-Parameter kann nur für Custom Web Publishing mit XSLT-Abfragen verwendet werden.

***string* in –token.[*string*] ist:** Alphanumerischer String beliebiger Länge, außer Leerzeichen, einschließlich der Ziffern 0-9, Kleinbuchstaben a-z oder Großbuchstaben A-Z.

Benutzerdefinierter Parameterwert ist: Beliebige Zeichenfolge, die URL-kodiert ist.

Optional für: Alle XSLT-Abfragen

Beispiel:

http://192.168.123.101/fmi/xsl/template/my_stylesheet.xsl?–db=employees&–lay=departments&–grammar=fmresultset&–token.D100=Active&–findall

Weitere Informationen finden Sie unter „Verwendung von Tokens zur Übergabe von Informationen zwischen Stylesheets“ auf Seite 65.

Anhang B
Fehlercodes für Custom Web Publishing

Die Web Publishing Engine unterstützt drei Arten von Fehlercodes, die für Custom Web Publishing auftreten können:

- **Datenbank- und Query-String-Fehler:** Die Web Publishing Engine generiert für eine veröffentlichte Datenbank jedes Mal einen Fehlercode, wenn eine XML-Datenabfrage auftritt. Weitere Informationen finden Sie im nächsten Abschnitt, „Fehlercodenummern für FileMaker-Datenbanken“.
- **Web Publishing Engine-Fehler:** Wenn sich die Web Publishing Engine im Entwicklungsmodus befindet, generiert sie eine spezifische Fehlerseite, falls in der Web Publishing Engine selbst ein Fehler auftritt. Im Produktionsmodus wird eine allgemeine Textmeldung angezeigt. Siehe „Fehlercodenummern für die Web Publishing Engine“ auf Seite 120.
- **Fehler in FileMaker XSLT-Erweiterungsfunktionen:** Sie können die Erweiterungsfunktion fmxslt:check_error_status() in einem XSLT-Stylesheet verwenden, um den Fehlerstatus von Erweiterungsfunktionen bei ihrem Aufruf zu überprüfen. Siehe

Fehlercodenummern für FileMaker-Datenbanken

Die Web Publishing Engine generiert für eine in XML-Format veröffentlichte Datenbank jedes Mal einen Fehlercode, wenn Daten angefordert werden. Diese Art von Fehlercodewert wird am Beginn des XML-Dokuments in das <error code>-Element für die fmresultset-Grammatik bzw. im <ERRORCODE>-Element für die Grammatik FMPXMLRESULT oder FMPDSORESULT eingefügt. Der Fehlercode 0 gibt an, dass kein Fehler aufgetreten ist.

Beispiel des Datenbankfehlercodes in der fmresultset-Grammatik:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE fmresultset PUBLIC "-//FMI//DTD fmresultset//EN" "/fmi/xml/fmresultset.dtd">
<fmresultset xmlns="http://www.filemaker.com/xml/fmresultset" version="1.0">
   <error code="0"></error>
```

Beispiel des Datenbankfehlercodes in der FMPXMLRESULT-Grammatik:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<!DOCTYPE FMPXMLRESULT PUBLIC "-//FMI//DTD FMPXMLRESULT//EN" "/fmi/xml/FMPXMLRESULT.dtd">
<fmpxmlresult xmlns="http://www.filemaker.com/fmpxmlresult">
   <ERRORCODE>0</ERRORCODE>
```

Als Entwickler der Custom Web Publishing-Lösung bleibt es Ihnen überlassen, den Wert des <error code>- bzw. <ERRORCODE>-Elements zu prüfen und geeignete Maßnahmen vorzusehen. Die Web Publishing Engine behandelt keine Datenbankfehler.

Fehlernummer	Beschreibung
-1	Unbekannter Fehler
0	Kein Fehler
1	Aktion durch Benutzer abgebrochen

Fehlernummer	Beschreibung
2	Speicherfehler
3	Befehl nicht verfügbar (z. B. falsches Betriebssystem, falscher Modus etc.)
4	Befehl unbekannt.
5	Befehl ungültig (z. B. ist für den Scriptschritt „Feldwert setzen“ keine Formel angegeben).
6	Datei ist schreibgeschützt.
7	Speicherüberlauf
8	Leeres Ergebnis
9	Ungenügende Berechtigungen
10	Angeforderte Daten fehlen.
11	Name ist nicht gültig.
12	Name existiert bereits.
13	Datei oder Objekt ist in Gebrauch.
14	Außerhalb des gültigen Bereichs
15	Teilen durch null nicht möglich.
16	Operation fehlgeschlagen, Anfrage wiederholen (z. B. eine Benutzeranfrage).
17	Konvertierung von fremdem Zeichensatz in UTF-16 fehlgeschlagen.
18	Client muss Kontoinformationen liefern, um fortzufahren.
19	Zeichenfolge enthält andere Zeichen als A-Z, a-z, 0-9 (ASCII).
100	Datei fehlt.
101	Datensatz fehlt.
102	Feld fehlt.
103	Beziehung fehlt.
104	Script fehlt.
105	Layout fehlt.
106	Tabelle fehlt.
107	Index fehlt.
108	Werteliste nicht vorhanden.
109	Berechtigung fehlt.
110	Bezugstabellen fehlen.
111	Feldwiederholung ist ungültig.
112	Fenster fehlt.
113	Funktion fehlt.
114	Dateiverweis fehlt.
130	Dateien sind beschädigt oder fehlen und müssen neu installiert werden.
131	Dateien des Sprachpakets fehlen (z. B. Vorlagendateien).

Fehlernummer	Beschreibung
200	Zugriff auf Datensatz verweigert.
201	Feld kann nicht geändert werden.
202	Zugriff auf Feld verweigert.
203	Keine zu druckenden Datensätze in der Datei bzw. Passwort erlaubt kein Drucken.
204	Kein Zugriff auf Feld(er) in Sortierfolge
205	Benutzer hat keine Zugriffsrechte, um neue Datensätze zu erstellen; Import überschreibt bestehende Daten.
206	Benutzer hat keine Zugriffsrechte, um das Passwort zu ändern, oder die Datei ist schreibgeschützt.
207	Benutzer hat nicht genügend Zugriffsrechte, um das Datenbankschema zu ändern, oder die Datei ist schreibgeschützt.
208	Passwort enthält zu wenige Zeichen.
209	Neues Passwort muss sich vom bestehenden unterscheiden.
210	Benutzerkonto ist inaktiv.
211	Passwort ist abgelaufen.
212	Ungültiges Benutzerkonto und/oder Passwort. Versuchen Sie es erneut.
213	Benutzerkonto und/oder Passwort existiert nicht.
214	Zu viele Anmeldeversuche
215	Administratorrechte können nicht dupliziert werden.
216	Gastkonto kann nicht dupliziert werden.
217	Benutzer hat nicht genügend Zugriffsrechte, um Administratorkonto zu ändern.
300	Datei ist geschützt oder in Gebrauch.
301	Datei ist blockiert durch anderen Anwender.
302	Tabelle ist blockiert durch anderen Anwender.
303	Datenbankschema ist blockiert durch anderen Anwender.
304	Layout ist blockiert durch anderen Anwender.
306	Datensatzänderungs-ID stimmt nicht überein.
400	Suchkriterien sind leer.
401	Kein Datensatz entspricht der Abfrage.
402	Kein Vergleichsfeld für eine Referenz
403	Maximales Datensatzlimit für FileMaker Pro-Demo wird überschritten.
404	Ungültige Sortierfolge
405	Angegebene Datensatzzahl übersteigt die Anzahl der ausschließbaren Datensätze.
406	Ungültige Kriterien für Ersetzen/Neunummerierung
407	Ein oder beide Gruppierfeld(er) fehlen (ungültige Beziehung).
408	Angegebenes Feld hat ein Datenformat, das diesem Befehl nicht entspricht.

Fehlernummer	Beschreibung
409	Ungültige Importfolge
410	Ungültige Exportfolge
412	Falsche Version von FileMaker Pro verwendet, um die Datei wiederherzustellen
413	Angegebenes Feld hat ungeeigneten Feldtyp.
414	Layout kann das Ergebnis nicht anzeigen.
415	Ein oder mehrere erforderliche Bezugsdatensätze sind nicht verfügbar.
500	Datumswert entspricht nicht den Überprüfungskriterien.
501	Zeitwert entspricht nicht den Überprüfungskriterien.
502	Zahlenwert entspricht nicht den Überprüfungskriterien.
503	Feldwert entspricht nicht der Bereichsüberprüfung.
504	Feldwert entspricht nicht der Eindeutigkeitsüberprüfung.
505	Feldwert existiert nicht in der Datenbank und entspricht nicht der Existenzüberprüfung.
506	Feldwert entspricht nicht der Überprüfung nach Bestandteil einer Werteliste.
507	Feldwert entspricht nicht der Überprüfung durch Berechnung.
508	Ungültiger Wert wurde in Suchenmodus eingegeben.
509	Feld verlangt gültigen Wert.
510	Bezugswert ist leer oder nicht verfügbar.
511	Der Wert im Feld überschreitet die maximale Anzahl der zulässigen Zeichen.
600	Druckerfehler aufgetreten.
601	Kombination von Kopf- und Fußbereich übersteigt eine Seitenlänge.
602	Datenbereich passt für aktuelle Spalteneinstellung nicht auf eine Seite.
603	Verbindung zum Drucker getrennt.
700	Datei hat falschen Dateityp für Import.
706	EPSF-Datei hat keine Übersichtsgrafik.
707	Grafikfilter nicht vorhanden.
708	Dateiimport nicht möglich bzw. Farbmonitor für Import erforderlich.
709	Import des QuickTime-Films fehlgeschlagen.
710	QuickTime-Dateiverweis konnte nicht aktualisiert werden, da für Datenbankdatei nur Lesezugriff besteht.
711	Importfilter nicht vorhanden.
714	Zugriffsrechte reichen für diesen Befehl nicht aus.
715	Benannter Bereich oder Tabellenblatt von Excel konnte nicht gefunden werden.
716	Eine SQL Anfrage mit DELETE, INSERT oder UPDATE ist nicht für ODBC Import erlaubt.
717	Zum Fortsetzen des Imports bzw. Exports sind nicht genügend XML/XSL-Daten vorhanden.
718	XML-Parsingfehler (von Xerces)

Fehlernummer	Beschreibung
719	Fehler beim Transformieren von XML mit XSL (von Xalan)
720	Fehler beim Export: Das vorgesehene Format unterstützt keine Wiederholfelder.
721	Im Parser oder Transformer ist ein unbekannter Fehler aufgetreten.
722	Daten können nicht in eine Datei importiert werden, die keine Felder hat.
723	Sie sind nicht berechtigt, Datensätze in der Zieltabelle hinzuzufügen oder zu ändern.
724	Sie sind nicht berechtigt, Datensätze in der Zieltabelle hinzuzufügen.
725	Sie sind nicht berechtigt, Datensätze in der Zieltabelle zu ändern.
726	In der Importdatei sind mehr Datensätze vorhanden als in der Zieltabelle. Nicht alle Datensätze werden importiert.
727	In der Zieltabelle sind mehr Datensätze vorhanden als in der Importdatei. Nicht alle Datensätze werden aktualisiert.
729	Fehler beim Import. Datensätze konnten nicht importiert werden.
730	Nicht unterstützte Excel-Version. (Konvertieren Sie die Datei in Excel 7.0 (Excel 95), Excel 97, 2000 oder XP und versuchen Sie es erneut.)
731	Die für den Import ausgewählte Datei enthält keine Daten.
732	Diese Datei kann nicht eingefügt werden, weil sie selbst weitere Dateien enthält.
733	Eine Tabelle kann nicht in sich selbst importiert werden.
734	Dieser Dateityp kann nicht als Bild dargestellt werden.
735	Dieser Dateityp kann nicht als Bild dargestellt werden. Er wird eingefügt und dann als Datei dargestellt.
800	Datei konnte auf Datenträger nicht erstellt werden.
801	Temporärdatei konnte auf Systemdatenträger nicht erstellt werden.
802	Datei konnte nicht geöffnet werden.
803	Datei in Einzelbenutzer-Status oder Host nicht vorhanden.
804	Datei konnte in ihrem aktuellen Status nicht mit Nur-Lese-Zugriff geöffnet werden.
805	Datei ist beschädigt; stellen Sie die Datei wieder her.
806	Datei kann mit dieser Version von FileMaker Pro nicht geöffnet werden.
807	Datei ist keine FileMaker Pro-Datei oder ist schwer beschädigt.
808	Datei kann wegen beschädigter Zugriffsrechte nicht geöffnet werden.
809	Datenträger voll
810	Datenträger fixiert
811	Temporärdatei kann nicht als FileMaker Pro-Datei geöffnet werden.
813	Fehler bei Datensatz-Synchronisation im Netzwerk
814	Datei(en) kann (können) nicht geöffnet werden, da die maximale Anzahl an Dateien geöffnet ist.
815	Referenzdatei konnte nicht geöffnet werden.
816	Datei konnte nicht konvertiert werden.

Fehlernummer	Beschreibung
817	Der Bindungsschlüssel der Datei stimmt nicht mit dieser Laufzeitanwendung überein.
819	Lokale Kopie einer remote Datei kann nicht gespeichert werden.
820	Datei wird geschlossen.
821	Host hat die Verbindung getrennt.
822	FMI-Dateien nicht gefunden; fehlende Dateien neu installieren.
823	Datei kann nicht auf Einzelbenutzer gesetzt werden, da Gäste verbunden sind.
824	Datei ist beschädigt oder keine FileMaker-Datei.
900	Allgemeiner Fehler in der Rechtschreibprüfung
901	Standardwörterbuch nicht installiert.
902	Hilfe-System konnte nicht gestartet werden.
903	Befehl kann nicht in einer gemeinsam genutzten Datei verwendet werden.
904	Befehl kann nur in einer Datei verwendet werden, die von FileMaker Pro Server freigegeben wurde.
905	Kein aktives Feld ausgewählt, Befehl kann nur mit aktivem Feld verwendet werden.
920	Initialisierung der Rechtschreibprüfung nicht möglich.
921	Anwenderwörterbuch kann nicht zur Bearbeitung geladen werden.
922	Anwenderwörterbuch existiert nicht.
923	Anwenderwörterbuch ist schreibgeschützt.
951	Ein unerwarteter Fehler ist aufgetreten.
954	Nicht unterstützte XML-Grammatik
955	Kein Datenbankname
956	Maximale Anzahl von Datenbanksitzungen überschritten.
957	Widersprüchliche Befehle
958	Parameter fehlt in Query
1200	Generischer Rechenfehler
1201	In dieser Funktion gibt es zu wenige Parameter.
1202	In dieser Funktion gibt es zu viele Parameter.
1203	Unerwartetes Ende der Berechnung
1204	Es wird eine Zahl, eine Textkonstante, ein Feldname bzw. „(“ erwartet.
1205	Kommentar ist nicht mit „*/“ beendet.
1206	Textkonstante muss mit einem Anführungszeichen enden.
1207	Klammer unvollständig
1208	Operator fehlt, Funktion nicht gefunden oder „(“ nicht erwartet.
1209	Name (z. B. Feldname oder Layoutname) fehlt.
1210	Plugin-Funktion wurde bereits registriert.

Fehlernummer	Beschreibung
1211	In dieser Funktion ist die Listennutzung nicht zulässig.
1212	Hier wird ein Operator (z. B. +, -, *) erwartet.
1213	Diese Variable wurde bereits in der SetzeVars-Funktion definiert.
1214	MITTELWERT, ANZAHL, ERWEITERN, HOLEWIEDERHOLFELDWERT, MAX, MIN, NBW, STABW, SUMME und HOLEERGEBNISWERT: Ausdruck gefunden, wo nur ein Feld benötigt wird.
1215	Dieser Parameter ist ein ungültiger Statusfunktionsparameter.
1216	Als erstes Argument einer HOLEERGEBNISWERT-Funktion können nur Statistikfelder angegeben werden.
1217	Gruppierfeld ist ungültig.
1218	Zahl kann nicht berechnet werden.
1219	Ein Feld kann nicht in seiner eigenen Formel benutzt werden.
1220	Feldtyp muss normal oder berechnet sein.
1221	Datentyp muss Zahl, Datum, Zeit oder Zeitstempel sein.
1222	Formel kann nicht gespeichert werden.
1223	Die angegebene Funktion existiert nicht.
1400	ODBC-Client-Treiberinitialisierung fehlgeschlagen. Stellen Sie sicher, dass die ODBC-Client-Treiber richtig installiert sind. **Hinweis** Die Plugin-Komponente für das Bereitstellen von Daten über ODBC wird automatisch zusammen mit FileMaker Server installiert. Die ODBC-Client-Treiber werden über die FileMaker Server Web Publishing-CD installiert. Weitere Informationen finden Sie in *Installation der FileMaker ODBC- und JDBC-Client-Treiber*.
1401	Umgebung konnte nicht zugeteilt werden (ODBC).
1402	Umgebung konnte nicht freigegeben werden (ODBC).
1403	Verbindung trennen fehlerhaft (ODBC).
1404	Verbindung konnte nicht zugeteilt werden (ODBC).
1405	Verbindung konnte nicht freigegeben werden (ODBC).
1406	Überprüfung von SQL API fehlgeschlagen (ODBC).
1407	Anweisung konnte nicht zugeteilt werden (ODBC).
1408	Erweiterter Fehler (ODBC)

Fehlercodenummern für die Web Publishing Engine

Wenn sich die Web Publishing Engine im Entwicklungsmodus befindet, generiert sie eine spezifische Fehlerseite, sobald in der Web Publishing Engine selbst ein Fehler auftritt. Dieser Fehlertyp kann eine Vielzahl von Ursachen haben, z. B. wenn die Web Publishing Engine zu Folgendem nicht in der Lage ist:

- Finden einer angeforderten oder verschachtelten (über <xsl:include>) Stylesheet-Datei
- Parsen einer angeforderten oder verschachtelten Stylesheet-Datei aufgrund eines XML-Fehlers in der Datei
- Generieren eines Stylesheets aus der Datei aufgrund eines XSLT- oder XPath-Fehlers in der Datei
- Verarbeiten der Abfrage, da die XML-Grammatik im CGI nicht korrekt angegeben ist
- Kommunizieren mit dem Web Publishing Core, um XML abzurufen

Wenn die Web Publishing Engine im Entwicklungsmodus läuft, enthält die Fehlerseite für diesen Fehlertyp eine Fehlermeldung und eine eingeklammerte Fehlernummer. Beispiel:

FileMaker Custom Web Publishing mit XSLT
Die abgefragte Stylesheet-Datei wurde nicht gefunden. (FILE-ER0001)
Datei: home4.xsl
Zeile:
Spalte:
Hinweis: Wenn eine Quelldatei und eine Zeilennummer angegeben sind, handelt es sich hierbei um die Stelle, an der der Fehler entdeckt wurde. Dies muss aber nicht die genaue Stelle des Problems sein.

Beispiel für Fehlerseite, wenn sich die Web Publishing Engine im Entwicklungsmodus befindet

Die folgende Liste führt einige der Web Publishing Engine-Fehlercodewerte auf:

Fehlercode-Wert	Beschreibung
QUERY-ER0001	Im Query-Parameter –grammar ist keine XML-Grammatik angegeben.
QUERY-ER0002	„xxx“ ist keine gültige XML-Grammatik für FileMaker XSLT.
FILE-ER0001	Die abgefragte Stylesheet-Datei wurde nicht gefunden.
FILE-ER0002	Die angeforderte Datei wurde nicht gefunden.
UNKNOWN	Ein unerwarteter Fehler ist aufgetreten.
MCS-000 bis MCS-600	Ein unerwarteter Fehler ist aufgetreten.
MCS-601	Die Ressource „x“ konnte nicht geladen werden, weil Ressourcen dieses Typs nicht unterstützt werden: „x“
MCS-602	Der URL „x“ konnte nicht aufgelöst werden.
MCS-603	Die HTTP-Abfrage für „x“ gab einen Fehler des Typs „x“ zurück.
MCS-604	Die Ressource „x“ konnte nicht geladen werden, weil ein unerwarteter Fehler aufgetreten ist.
MCS-605	Die Ressource „x“ konnte nicht geladen werden, weil der Content-Typ ungültig war.

Fehlercode-Wert	Beschreibung
MCS-606	Die Ressource „x“ konnte nicht geladen werden, weil das Dokument einen XML-Fehler aufweist.
MCS-607	Die Ressource „x“ konnte aufgrund eines Authentifizierungsproblems nicht geladen werden.
MCS-700	Ein unerwarteter Fehler ist aufgetreten.
MCS-800	Ein unerwarteter Fehler ist aufgetreten.

Wenn sich die Web Publishing Engine im Produktionsmodus befindet, wird für Fehler in der Web Publishing Engine die folgende allgemeine Textmeldung auf der Seite pe_server_error.html angezeigt:

Bei der Verwendung von FileMaker Custom Web Publishing mit XSLT ist ein unerwarteter Fehler aufgetreten.

Die Standarddatei pe_server_error.html enthält die vorangehende Textmeldung in sechs Sprachen.

Als Entwickler können Sie den Text der Fehlerseite pe_server_error.html wie für Ihre Lösung erforderlich bearbeiten. Die Datei pe_server_error.html befindet sich im Ordner cwpe innerhalb des publishing-engine-Ordners auf dem Host, auf dem Sie die Web Publishing Engine installiert haben.

Informationen über das Setzen der Web Publishing Engine in den Entwicklungs- oder Produktionsmodus finden Sie im Handbuch *FileMaker Server Advanced Web Publishing Installation*.

Fehlercodenummern für die FileMaker XSLT-Erweiterungsfunktionen

Die Erweiterungsfunktion fmxslt:check_error_status() (siehe „Prüfen des Fehlerstatus von Erweiterungsfunktionen“ auf Seite 85) gibt einen der Fehler in der folgenden Tabelle zurück:

Fehlercode-Wert	Beschreibung
-1	Unbekannter Fehler
0	Kein Fehler
	Allgemeine Fehler
10000	Ungültiger Header-Name
10001	Ungültiger HTTP-Statuscode
	Sitzungsfehler
10100	Unbekannter Sitzungsfehler
10101	Angeforderter Sitzungsname bereits in Gebrauch.
10102	Zugriff auf Sitzung nicht möglich – sie existiert evtl. nicht.
10103	Zeitüberschreitung der Sitzung
10104	Angegebenes Sitzungsobjekt existiert nicht.

Fehlercode-Wert	Beschreibung
10200	Unbekannter Nachrichtenfehler
10201	Nachrichten-Formatfehler
10202	SMTP-Feldfehler in Nachricht
10203	Fehler in „To-Feld“ der Nachricht
10204	Fehler in „From-Feld“" der Nachricht
10205	Fehler in „CC-Feld“ der Nachricht
10206	Fehler in „BCC-Feld“ der Nachricht
10207	Fehler in „Subject-Feld“ der Nachricht
10208	Fehler in „Reply-To-Feld“ der Nachricht
10209	Nachrichten-Textfehler
10210	Rekursiver Mail-Fehler – Aufruf von send_email() in E-Mail-XSLT-Stylesheet versucht
10211	SMTP-Authentifizierungsfehler – Anmeldung fehlgeschlagen oder falscher Authentifizierungstyp angegeben
10212	Unzulässige Funktionsverwendung – Aufruf von call set_header(), set_status_code() oder set_cookie() in E-Mail-XSLT-Stylesheet versucht
10213	SMTP-Server ist ungültig oder funktioniert nicht.

Fehlercode-Wert	Beschreibung
	Formatierungsfehler
10300	Unbekannter Formatierungsfehler
10301	Ungültiges Datum-Zeit-Format
10302	Ungültiges Datumsformat
10303	Ungültiges Zeitformat
10304	Ungültiges Tagesformat
10305	Falsch formatierter Datum-Zeit-String
10306	Falsch formatierter Datums-String
10307	Falsch formatierter Zeit-String
10308	Falsch formatierter Tages-String
10309	Nicht unterstützte URL-Textkodierung
10310	Ungültige URL-Textkodierung
10311	Musterfehler in regulärem Ausdruck

Anhang C
Konvertieren von CDML-Lösungen in FileMaker XSLT

Dieser Anhang erläutert die Ergebnisse der Verwendung von FileMaker CDML Converter zur Konvertierung von CDML-Formatdateien in FileMaker XSLT-Stylesheets. Weitere Informationen über die Verwendung von FileMaker CDML Converter finden Sie unter „Verwenden von FileMaker CDML Converter“ auf Seite 50

Hinweis In diesem Anhang und in den generierten Kommentaren in konvertierten XSLT-Stylesheets wird der Begriff „XSLT-CWP“ für FileMaker Custom Web Publishing mit XSLT verwendet.

Verfahren bei der Konvertierung von CDML-Lösungen in FileMaker XSLT-Lösungen

Der CDML Converter geht wie folgt vor, um Dateien der CDML-Lösung im Quellordner zu konvertieren, umzubenennen oder zu kopieren:

- CDML-Formatdateien, die CDML-Tags oder Query-Parameter enthalten, werden in XSLT-Stylesheets konvertiert und dann im Zielordner gespeichert.
- Die Dateinamenerweiterungen von HTML-Dateien werden unabhängig davon, ob sie CDML-Tags enthalten, in .xsl umbenannt und dann in den Zielordner kopiert. Beispielsweise wird datei.html in datei.xsl umbenannt.
- Alle Großbuchstaben in Dateinamen werden in Kleinbuchstaben umgewandelt, da alle konvertierten Dateiverweise Kleinbuchstaben verwenden und XSLT Groß-/Kleinschreibung unterscheidet.
- Nicht-HTML-Dateien, die keine CDML-Tags oder Query-Parameter enthalten, werden ohne Änderungen in den Zielordner kopiert.
- Alle Ordner in der Quellordnerhierarchie werden automatisch im Zielordner angelegt.

Für die Konvertierung jeder CDML-Formatdatei in ein XSLT-Stylesheet führt der CDML Converter folgende Aktionen aus:

- entfernt das Tag <!DOCTYPE>, wenn es vorhanden ist.
- ordnet alle booleschen CDML-Ausdrücke XPath-Ausdrücken zu.
- konvertiert in jeder Instanz des CDML-Tags –format alle Dateinamenerweiterungen in Dateinamenverweisen zu .xsl.
- ordnet alle CDML-Ersetzungs-Tags und dazwischen liegende Parameter den XSLT-CWP-Anweisungen zu.
- fügt die Verarbeitungsanweisung <?xslt-cwp-query params="–grammar=fmresultset"?> in alle konvertierten Stylesheets ein, um die fmresultset-Grammatik anzugeben. Weitere Informationen finden Sie unter „Verwenden von statisch definierten Query-Befehlen und Query-Parametern“ auf Seite 62
- fügt einen Wert für das Kodierungsattribut in der Verarbeitungsanweisung <?xml ... ?> und dem <xsl:output>-Element ein, der der Textkodierungsoption entspricht, die Sie beim Konvertieren im CDML Converter gewählt haben.

- fügt den Wert xml für das Methodenattribut des <xsl:output>-Elements ein, wenn die Eingabedatei mit einer <?xml ... ?>-Verarbeitungsanweisung beginnt. Wenn die Eingabedatei eine HTML-Datei ist, wird html als Wert des Methodenattributs eingefügt; andernfalls wird text eingefügt.
- fügt die folgenden Deklarationen für die Namespaces, XSLT-Parameter und Variablen ein, die in einem Stylesheet verwendet werden. Die Deklarationen werden vor und nach den konvertierten XSLT-CWP-Anweisungen eingefügt:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0"
    xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
    xmlns:fmrs="http://www.filemaker.com/xml/fmresultset"
    xmlns:fml="http://www.filemaker.com/fmpxmllayout"
    xmlns:fmq="http://www.filemaker.com/xml/query"
    xmlns:fmxslt="xalan://com.fmi.xslt.ExtensionFunctions"
    xmlns:xalan="http://xml.apache.org/xalan"
    exclude-result-prefixes="xsl fmrs fml fmq fmxslt xalan">
    <xsl:param name="authenticated-xml-base-uri"/>
    <xsl:param name="request-query"/>
    <xsl:variable name="layout" select="document(concat($authenticated-xml-base-uri,
    '/fmi/xml/FMPXMLLAYOUT.xml?','–db=', /fmrs:fmresultset/fmrs:database/@name,'&amp;
    –lay=', /fmrs:fmresultset/fmrs:database/@layout, '&amp;–view='))"/>
    <xsl:variable name="current-find"><xsl:call-template name="get-current-find"/></xsl:variable>
    <xsl:variable name="current-sort"><xsl:call-template name="get-current-sort"/></xsl:variable>
    <xsl:variable name="current-action"><xsl:call-template name="get-current-action"/></xsl:variable>
    <xsl:variable name="current-lop"><xsl:call-template name="get-current-lop"/></xsl:variable>
    <xsl:variable name="current-max"><xsl:call-template name="get-current-max"/></xsl:variable>
    <xsl:variable name="current-skip"><xsl:call-template name="get-current-skip"/></xsl:variable>
    <xsl:include href="cdml2xsl_utilities.xsl"/>
    <xsl:output method="html" encoding="ISO-8859-1"/>
    <xsl:template match="/fmrs:fmresultset">
        ... die von CDML/HTML konvertierten XSLT-CWP-Anweisungen werden hier eingefügt ...
    </xsl:template>
</xsl:stylesheet>
```

Für alle Standard Generalized Markup Language (SGML)-Eingabedateien einschließlich XML- und HTML-Dateien fügt der CDML Converter wie erforderlich Zeilenumbrüche ein und rückt verschachtelte Elemente ein. Der CDML Converter konvertiert die Dateien auch durch folgende Aktionen in wohlgeformte XML-Dokumente:

- fügt fehlende End-Tags direkt vor dem End-Tag des übergeordneten Elements ein, außer für die folgenden leeren HTML-Elemente, die in HTML-Dokumenten wie gezeigt abgeschlossen werden: <area />, <base />, <basefont />, <br />, <col />, <frame />, <hr />, <img />, <input />, <isindex />, <link />, <meta />, <param />
- korrigiert alle End-Tags so, dass die Verschachtelung korrekt ist,
- ändert alle Element- und Attributnamen zu Kleinbuchstaben, ändert aber keine Attributwerte
- überprüft, ob alle Attributwerte von doppelten Hochkommas umgeben sind
- weist einem fehlenden Attributwert den Attributnamen zu, z. B. attribut_name="attribut_name"

Konvertierung von CDML-Aktions-Tags, Variablen-Tags und URLs

Zur Konvertierung von CDML-Aktions-Tags, Variablen-Tags und URLs führt der CDML Converter folgende Aktionen aus:

- Der CDML Converter ändert keine nicht identifizierten Tagnamen, denen ein Gedankenstrich (–) vorangeht. Nicht identifizierte Tagnamen in einem URL werden an das Ende des Query-Strings verschoben. Der CDML Converter nimmt an, dass Tagnamen, denen kein Gedankenstrich vorangeht, <fieldname>-Variablen-Tags sind, und lässt sie an der passenden Position im URL.
- Der CDML Converter sucht nach dem Textstring fmpro?, um URLs zu identifizieren. In Nicht-SGML-Dokumenten und SGML-Kommentaren identifiziert der CDML Converter URLs an jeder Position. In SGML-Elementen identifiziert der CDML Converter nur URLs, die sich in Element-Attributwerten befinden.
- Der CDML Converter identifiziert das Variablen-Tag –format nur dann außerhalb von URLs, wenn es sich in Eingabeelementen in HTML-Dokumenten befindet. Der CDML Converter ignoriert alle anderen Formularelemente wie Textauswahl- und Textbereich-Elemente.
- Der CDML Converter ersetzt alle <input>-Elemente, die das CDML-Tag –format mit XSLT-CWP-Anweisungen enthalten. Zum Beispiel wird das folgende <input>-Element und –format-Tag:

  <input type="hidden" name="-format" value="results.htm">

  ersetzt durch:

  <xsl:attribute name="action">results.xsl</xsl:attribute>

- Der CDML Converter konvertiert URLs in Abfragen und <form> <input>-Elementen in die neue Syntax für XSLT. Weitere Informationen finden Sie unter „Erläuterung der URL-Syntax für FileMaker XSLT-Stylesheets“ auf Seite 58.
- Der CDML Converter konvertiert URLs, die FileMaker XML-Daten anfordern, in denen –format den Wert –fmp_xml oder –dso_xml verwendet, in die neue URL-Syntax für XML-Daten.
  Weitere Informationen finden Sie unter „Erläuterung der URL-Syntax für XML-Daten und Medienobjekte“ auf Seite 28.

- Der CDML Converter konvertiert URLs, die FileMaker-Bilddaten (über –img) anfordern, in die URL-Syntax für das Anfordern von Medienobjekten in XSLT-Lösungen. Weitere Informationen finden Sie unter „Erläuterung der URL-Syntax für FileMaker-Medienobjekte in XSLT-Lösungen“ auf Seite 59. Wenn eine Query-Abfrage das img-key-Format enthält, konvertiert CDML Converter es auf dieselbe Weise wie das [FMP-Image]-Variablen-Tag. Zum Beispiel wird diese CDML-Abfrage:

  <img src="fmpro?–db=employees&key=[FMP-Field: fieldname, url]&–img=">

  ebenso wie diese konvertiert:

  <img src="[FMP-Image: fieldname]">

  Informationen über die Konvertierung von [FMP-Image] finden Sie unter „CDML-Tagname: Image (Bild)“ auf Seite 159.

- Der CDML Converter konvertiert Tagnamen zu XSLT-CWP-Query-Befehlen und Parameternamen mit Ausnahme des CDML-Variablen-Tags <fieldname> in Suchabfragen in Kleinbuchstaben. Werte für Query-Befehle und -Parameter werden nicht geändert.
- Der CDML Converter stellt die konvertierten XSLT-CWP-Query-Befehle und -Parameter in einem URL in die folgende Reihenfolge um: –db, –grammar, –lay und alle anderen Query-Parameter und zuletzt den Query-Befehl wie –findall. Der CDML Converter ändert nicht die Reihenfolge von <form> <input>-Elementen, die Query-Befehle und -Parameter enthalten.
- Wenn eine CDML-Abfrage für den Query-Befehl –find oder –findall nicht den Query-Parameter –max angibt, fügt der CDML Converter der konvertierten XSLT-Abfrage –max=25 hinzu. Der CDML Converter verwendet den Wert 25, um dasselbe Verhalten zu erzeugen wie die ursprüngliche CDML-Abfrage, in der 25 Datensätze zurückgegeben werden, wenn der Query-Parameter –max fehlt. Beachten Sie, dass sich dieses Verhalten in XML- oder XSLT-Abfragen geändert hat. Wenn der Query-Parameter –max in einer XML- oder XSLT-Abfrage nicht angegeben ist, werden alle Datensätze zurückgegeben. Weitere Informationen finden Sie unter „Query-Parameter –max (Maximum Datensätze)“ auf Seite 107.
- Bei einem Konvertierungsfehler fügt der CDML Converter einen Fehlerkommentar in das konvertierte Stylesheet direkt nach dem Element ein, bei dem der Fehler aufgetreten ist. Sie müssen Fehler im Stylesheet manuell korrigieren. Ein Fehlerkommentar hat folgendes Format:

  <!—CDML Converter-FEHLER: <Beschreibung des Fehlers> -->

- Bei einem Konvertierungsfehler fügt der CDML Converter einen Warnungskommentar direkt nach dem Element, bei dem die Ursache für die Warnung aufgetreten ist, in das konvertierte Stylesheet ein. Sie müssen im Stylesheet keine Warnungen korrigieren, aber Sie sollten sie untersuchen, um andere Fehler zu vermeiden. Ein Warnungskommentar hat folgendes Format:

  <!—CDML Converter-WARNUNG: <Beschreibung der Warnung> -->

- Der CDML Converter entfernt veraltete CDML-Aktions- und Variablen-Tags aus URLs und fügt einen Fehlerkommentar an der entsprechenden Stelle im Stylesheet ein. Weitere Informationen hierzu finden Sie unter „Konvertierung veralteter CDML-Aktions-Tags“ und appendix C, „Konvertierung veralteter CDML-Variablen-Tags“
- Der CDML Converter konvertiert nicht die veralteten –mail-Variablen-Tags (wie –mailto und –mailfrom). Weitere Informationen finden Sie unter „Konvertierung veralteter CDML-Aktions-Tags“ auf Seite 130. Um die E-Mail-Funktionalität der –mail-Variablen-Tags zu ersetzen, verwenden Sie eine fmxslt:send_email ()-Erweiterungsfunktion. Weitere Informationen finden Sie unter „Senden von E-Mail-Nachrichten von der Web Publishing Engine“ auf Seite 74.

- Wenn ein <form>-Element mit einem Aktionsparameter, der auf fmpro endet, kein -format <input>-Element enthält, fügt der CDML Converter den folgenden Fehlerkommentar ein:

  <!—CDML CONVERTER-FEHLER: Parameter '–format' nicht gefunden -->.

Konvertierung der CDML-Variablen-Tags –error und –errornum

Der CDML Converter konvertiert die CDML-Tags –error und –errornum in Token-Werte. In diesem Original-URL:

fmpro?–db=employees&–format=format.htm&–error=error.htm&–errnum=401&–view

wird z. B. das Tag –error in –token.error und das Tag –errnum in –token.errnum konvertiert. Der konvertierte XSLT-CWP-URL lautet:

format.xsl?–db=employees&–token.error=error.xsl&–token.errnum=401&–view

Das konvertierte Stylesheet format.xsl enthält die folgenden Anweisungen:

```
<xsl:include href="cdml2xsl_includes.xsl"/>
<xsl:variable name="_errorcode" select="/fmrs:fmresultset/fmrs:error/@code"/>
<xsl:variable name="token.error" select="$request-query/fmq:query/fmq:parameter[@name = '–token.error']"/>
<xsl:variable name="token.errnum" select="$request-query/fmq:query/fmq:parameter[@name = '–token.errnum']"/>
<xsl:choose>
    <xsl:when test="$token.error != '' and ($_errorcode = $token.errnum)">
        <xsl:call-template name="error"/>
    </xsl:when>
    <xsl:otherwise>
        [... hier konvertierter format.xsl-Code ...]
    </xsl:otherwise>
</xsl:choose>
```

Die Datei error.html wird in das Stylesheet error.xsl konvertiert, das eine benannte Vorlage enthält:

```
<xsl:template name="err">
    [... hier konvertierter error.xsl-Code ... ]
</xsl:template>
```

Der CDML Converter übernimmt alle Fehlerdateien in eine globale Fehlerdatei mit dem Namen cdml2xsl_includes.xsl. Die folgende Anweisung übernimmt die globale Fehlerdatei in alle Stylesheets mit Fehlerbehandlung:

```
<xsl:include href="error.xsl"/>
...
```

Falls während der Verarbeitung des Stylesheets format.xsl ein Fehler auftritt und der Fehlercode der Fehlernummer entspricht, die für den Query-Parameter –token.errnum angegeben ist, wird der Code von error.xsl ausgeführt, wenn format.xsl die Vorlage mit dem Namen „error“ aufruft. Dieses Verhalten entspricht der ursprünglichen CDML-Funktionalität von –error und –errornum.

Konvertierung veralteter CDML-Aktions-Tags

Der CDML Converter behandelt die Konvertierung der folgenden veralteten CDML-Aktions-Tags wie folgt:

Dieses veraltete CDML-Aktions-Tag	Konvertierung
-dbclose	Wird nach der ursprünglichen Aktions-Tag-Position durch diesen Kommentar ersetzt: <!—CDML Converter-FEHLER: '–dbclose' nicht von XSLT-CWP unterstützt. -->
–dbopen	Wird nach der ursprünglichen Aktions-Tag-Position durch diesen Kommentar ersetzt: <!—CDML Converter-FEHLER: '–dbopen' nicht von XSLT-CWP unterstützt. -->
–img	Wird entfernt und der angegebene URL für das Bild wird in die neue URL-Syntax für Medienobjekte umgewandelt. Weitere Informationen hierzu finden Sie unter „Erläuterung der URL-Syntax für FileMaker-Medienobjekte in XML-Lösungen“ und „Erläuterung der URL-Syntax für FileMaker-Medienobjekte in XSLT-Lösungen“ auf Seite 59.

Konvertierung unterstützter CDML-Aktions-Tags

Der CDML Converter behandelt die Konvertierung der folgenden unterstützten CDML-Aktions-Tags wie folgt:

CDML-Aktions-Tag	Konvertierung in XSLT-CWP-Query-Parameter ohne Änderung in Syntax
–delete	–delete
–dup	–dup
–edit	–edit
–find	–find
–findall	–findall
–findany	–findany
–new	–new
–view	–view
–dbnames	–dbnames
–layoutnames	–layoutnames
–scriptnames	–scriptnames

Konvertierung veralteter CDML-Variablen-Tags

Der CDML Converter behandelt die Konvertierung der folgenden veralteten CDML-Variablen-Tags wie folgt:

Dieses veraltete CDML-Variablen-Tag	Wird nach der ursprünglichen Variablen-Tag-Position durch diesen Kommentar ersetzt
–errorfmtfield	<!—CDML Converter-FEHLER: '–errorfmtfield' wird nicht von XSLT-CWP unterstützt. -->
–fmtfield	<!—CDML Converter-FEHLER: '–fmtfield' wird nicht von XSLT-CWP unterstützt. -->
–format	Es wird kein Kommentar eingefügt. Stattdessen wird der URL zur Formatdatei in die neue URL-Syntax konvertiert, um auf den konvertierten XSLT-Stylesheet-Namen zu verweisen.
–mailbcc	<!—CDML Converter-FEHLER: '–mailbcc' wird nicht von XSLT-CWP unterstützt. Verwenden Sie die Funktion fmxslt:send_email(). -->
–mailcc	<!—CDML Converter-FEHLER: '–mailcc' wird nicht von XSLT-CWP unterstützt. Verwenden Sie die Funktion fmxslt:send_email(). -->
–mailfmtfield	<!—CDML Converter-FEHLER: '–mailfmtfield' wird nicht von XSLT-CWP unterstützt. Verwenden Sie die Funktion fmxslt:send_email(). -->
–mailformat	<!—CDML Converter-FEHLER: '–mailformat' wird nicht von XSLT-CWP unterstützt. Verwenden Sie die Funktion fmxslt:send_email(). -->
–mailfrom	<!—CDML Converter-FEHLER: '–mailfrom' wird nicht von XSLT-CWP unterstützt. Verwenden Sie die Funktion fmxslt:send_email(). -->
–mailhost	<!—CDML Converter-FEHLER: '–mailhost' wird nicht von XSLT-CWP unterstützt. Verwenden Sie die Funktion fmxslt:send_email(). -->
–mailsub	<!—CDML Converter-FEHLER: '–mailsub' wird nicht von XSLT-CWP unterstützt. Verwenden Sie die Funktion fmxslt:send_email(). -->
–mailto	<!—CDML Converter-FEHLER: '–mailto' wird nicht von XSLT-CWP unterstützt. Verwenden Sie die Funktion fmxslt:send_email(). -->
–password	<!—CDML Converter-FEHLER: '–password' wird nicht von XSLT-CWP unterstützt. -->

Konvertierung unterstützter CDML-Variablen-Tags

Hinweis Der CDML Converter behandelt die Konvertierung der folgenden unterstützten CDML-Variablen-Tags wie folgt:

CDML-Variablen-Tag	Wird in den folgenden XSLT-CWP-Query-Parameter umgewandelt:	Beschreibung
–db	–db	Keine Syntaxänderung. Alle Instanzen der .fp5-Dateinamenerweiterung werden entfernt.
–lay	–lay	Keine Syntaxänderung
–lop	–lop	Keine Syntaxänderung. Schlüsselwörter and und or werden in Kleinbuchstaben geändert.
–max	–max	Keine Syntaxänderung. Wenn –max nicht in CDML angegeben war, fügt der CDML Converter –max=25 in konvertiertes XSLT ein.
–modid	–modid	Keine Syntaxänderung

CDML-Variablen-Tag	Wird in den folgenden XSLT-CWP-Query-Parameter umgewandelt:	Beschreibung
–op	field.op	Wird so geändert, dass die neue Syntax für Suchoperatoren (field.op) unterstützt wird. Weitere Informationen finden Sie unter „Query-Parameter fieldname.op (Vergleichsoperator)“ auf Seite 105.
–recid	–recid	Keine Syntaxänderung
–script	–script	Dieser Kommentar wird nach dem Element eingefügt: <!—CDML2XSLT-WARNUNG: Bitte '–script'-Funktionalität mit XSLT-CWP überprüfen. -->
–script.prefind	–script.prefind	Keine Syntaxänderung. Prüfen Sie die '-script.prefind'-Funktionalität in XSLT-CWP.
–script.presort	–script.presort	Keine Syntaxänderung. Prüfen Sie die '–script.presort'-Funktionalität in XSLT-CWP.
–skip	–skip	Keine Syntaxänderung
–sortfield	–sortfield	Wird so geändert, dass die neue Syntax für Sortieroperatoren (–sortierfeld.vorrang) unterstützt wird. Weitere Informationen finden Sie unter „Query-Parameter –sortfield (Sortierfeld)“ auf Seite 110.
–sortorder	–sortorder	Wird so geändert, dass die neue Syntax für Sortieroperatoren (–sortorder.precedence) unterstützt wird. Weitere Informationen finden Sie unter „Query-Parameter –sortorder (Sortierfolge)“ auf Seite 110.
–styletype	–styletype	Keine Syntaxänderung
–stylehref	–stylehref	Keine Syntaxänderung
–token	–token	Keine Syntaxänderung
–token.ZAHL	–token.ZAHL	Keine Syntaxänderung
<Fieldname>	<Fieldname>	Keine Syntaxänderung

Konvertierung von booleschen CDML-Parametern in boolesche XPath-Parameter

Boolesche CDML-Ausdrücke, die in den Intratag-Parametern [FMP-If] und [FMP-ElseIf] verwendet werden, bestehen aus Parametern und Operatoren, die der CDML Converter in XPath-Ausdrücke konvertiert.

Boolescher CDML-Parameter	Wird konvertiert in diesen booleschen XPath-Operator	Kommentare
CanDelete	false()	Aus Sicherheitsgründen wird dieser Parameter in false() konvertiert, aber er wird nicht unterstützt. Dieser Kommentar wird nach dem Tag eingefügt: <!—CDML Converter-FEHLER: 'CanDelete' wird nicht von XSLT-CWP unterstützt. -->
CanEdit	false()	Aus Sicherheitsgründen wird dieser Parameter in false() konvertiert, aber er wird nicht unterstützt. Dieser Kommentar wird nach dem Tag eingefügt: <!—CDML Converter-FEHLER: 'CanEdit' wird nicht von XSLT-CWP unterstützt. -->
CanNew	false()	Aus Sicherheitsgründen wird dieser Parameter in false() konvertiert, aber er wird nicht unterstützt. Dieser Kommentar wird nach dem Tag eingefügt: <!—CDML Converter-FEHLER: 'CanNew' wird nicht von XSLT-CWP unterstützt. -->
IsSorted	boolean($request–query/fmq:query/ fmq:parameter[starts-with(@name, '–sortfield')]) = true())	
True	true()	
False	false()	
Checked	false()	
String	'String'	
Zahl	Zahl	
<fehlender Parameter>	''	[FMP-If xyz .eq.]...[/FMP-If] wird geändert zu: <xsl:choose><xsl:when test="'xyz' = ''"> ... </xsl:when> </xsl:choose>

Hinweis Wenn die Intratag-Parameter CurrentDate, CurrentDay oder CurrentTime in einem booleschen Ausdruck verwendet werden, ersetzt sie der CDML Converter durch die geeignete XSLT-CWP-Erweiterungsfunktion für Datumsvergleich wie fmxslt:compare_date (). Weitere Informationen finden Sie unter „Verwenden der Datums-, Zeit- und Tages-Erweiterungsfunktionen“ auf Seite 81.

Konvertierung von booleschen CDML-Operatoren in XPath

Boolescher CDML-Operator	Wird konvertiert in diesen booleschen XPath-Operator
x .and. y	x and y
x .or. y	x or y
x .xor. y	(x or y) and not (x and y)
x .eq. y	x = y
x .neq. y	x != y
x .gt. y	x > y
x .gte. y	x >= y
x .lt. y	x < y
x .lte. y	x <= y
x .cn. y	contains(x, y)
x .ncn. y	not (contains(x, y))

Konvertierung von CDML-Intratag-Parametern in XSLT-CWP

CDML-Intratag-Parameter sind CDML-Ersetzungs-Tags, die innerhalb eines Parameters eines anderen Ersetzungs-Tags verwendet werden. Wenn ein Intratag-Parameter als der Parameterwert benutzt wird, wird er von geschweiften Klammern umgeben. Beispiel:

```
–db={CurrentDatabase}
```

anstelle von:

```
[FMP-CurrentDatabase]
```

Der CDML Converter konvertiert Intratag-Parameter auf die gleiche Weise wie Ersetzungs-Tags. Weitere Informationen finden Sie unter „Konvertierung von CDML-Ersetzungs-Tags in XSLT-CWP“ auf Seite 139. Das folgende Beispiel zeigt, wie der CDML Converter die Konvertierung von Intratag-Parametern durchführt, die als Parameterwerte benutzt werden.

Beispiel für Konvertierung von Intratag-Parametern innerhalb Parameterwerten (in geschweiften Klammern angegeben):

Original-CDML:

```
<!-- Log page hits in another database -->
[FMP-InlineAction: –db=log.fp5, –lay=web, time="{CurrentTime}", date="{CurrentDate}",
page="This page!", browser="{ClientType}", ip="{ClientIP}", –new] [/FMP-InlineRequest]
```

Konvertierte XSLT-CWP:

```
<!-- Log page hits in another database -->
<xsl:variable select="document(concat('/fmi/xml/fmresultset.xml?–db=', fmxslt:url_encode('log'),
'&–lay=', fmxslt:url_encode('web'), '&time=', fmxslt:url_encode(fmxslt:get_time()), '&date=',
fmxslt:url_encode(fmxslt:get_date()), '&page=', fmxslt:url_encode('This Page!'), '&browser=',
fmxslt:url_encode(fmxslt:get_header('User-Agent')), '&ip=', fmxslt:url_encode($client-ip),
'&–new='))" />
```

Manuelle Korrektur von CDML-Konvertierungsfehlern

In einigen Situationen kann der CDML Converter die korrekte Konvertierung von CDML zu XSLT nicht automatisch ermitteln. In diesen Fällen müssen Sie mithilfe eines Texteditors oder XSLT-Stylesheet-Editors die Probleme in den konvertierten XSLT-Stylesheets manuell beheben:

- In FileMaker Pro sind Scriptnamen auf 100 Zeichen beschränkt. Bei der Datenbankkonvertierung werden alle längeren Scriptnamen in der Datenbank abgeschnitten. Da der CDML Converter keine Scriptnamen über dieser Grenze abschneidet, müssen Sie die Scriptnamen im XSLT-Stylesheet an die abgeschnittenen Namen der Datenbank anpassen.
- Der CDML Converter verwendet unabhängig von den regionalen Einstellungen des Hostcomputers, auf dem die Web Publishing Engine installiert ist, das Format „fm“, wenn er Server-seitige Funktionen in die XSLT-Stylesheets einfügt (z. B. fmxslt:get_date() und fmxslt:get_time()). Die „fm“-Formate sind MM/tt/jjjj für Datum, HH:mm:ss für Zeit und MM/tt/jjjj HH:mm:ss für Zeitstempel. Weitere Informationen finden Sie unter „Verwenden der Datums-, Zeit- und Tages-Erweiterungsfunktionen“ auf Seite 81 Wenn Ihre CDML-Formatdateien Datums- und Zeitformatstrings enthalten, die an diese Funktionen übergeben werden, müssen Sie die Strings nach der Konvertierung manuell ändern (z. B. „Jahr.Monat.Tag“ in „Tag.Monat.Jahr“). Um ein anderes oder bevorzugtes Format für die Ausgabewerte zu verwenden, verwenden Sie Berechnungsfunktionen oder JavaScript.
- Wenn eine CDML-Lösung ein Formular enthält, das dem Benutzer Werte für das Tag –format zur Auswahl bietet (z. B. mehrere Formatdateien), kann der CMDL Converter nicht automatisch eine XSLT-Lösung mit entsprechender Funktionalität erstellen. Sie müssen die konvertierte XSLT-Lösung manuell bearbeiten und eine separate Abfrage für jedes XSLT-Stylesheet einfügen, das Sie Web-Benutzern anbieten wollen.
- Wenn Query-Abfragen in Ihrer CDML-Lösung über das Tag –lay ein Layout angeben, stellen Sie sicher, dass die angegebenen Layouts alle Felder enthalten, auf die in den Abfragen verwiesen wird. Wenn das angegebene Layout nicht alle diese Felder enthält, müssen Sie entweder dem Layout die Felder hinzufügen oder den Query-Parameter –lay manuell so ändern, dass er auf ein Layout verweist, das alle Felder enthält. Alternativ können Sie mit dem Query-Parameter –lay.response das Layout wechseln, wenn Sie eine Abfrage mit einem Layout senden, dessen Felder in einem anderen Layout nicht vorhanden sind. Weitere Informationen finden Sie unter „Wechseln des Layouts für eine XML-Antwort“ auf Seite 42.
- Wenn Ihre CDML-Lösung Query-Abfragen enthält, die zur Angabe eines Layouts nicht das Tag –lay verwenden, fügt der CDML Converter den Abfragen automatisch den Query-Parameter –lay hinzu und gibt den Wert „AllFieldsLayout“ an. Sie müssen den Wert für den Parameter -lay entweder manuell in den konvertierten Stylesheets zum gewünschten Layout in Ihrer Datenbank ändern oder ein Layout mit dem Namen „AllFieldsLayout“ in Ihre Datenbank einfügen.
- Wenn Ihre CDML-Lösung die Variablen-Tags –script, –script.prefind oder –script.presort enthält, prüfen Sie die Scriptfunktionalität im konvertierten XSLT-Stylesheet.

- In CDML unterscheiden Vergleiche von Feld- und Datenbanknamen keine Groß-/Kleinschreibung, daher konnten Sie ein Tag wie [FMP-Field:myfield] verwenden, um auf ein Feld mit dem Namen MyField oder myField zu verweisen. In XSLT-CWP wird die Groß-/Kleinschreibung beim Vergleichen von Feld- und Datenbanknamen unterschieden, wenn sie nicht in einem Query-String benutzt werden. In den konvertierten Stylesheets müssen Sie etwaige Feld- und Datenbanknamen in XSLT-Anweisungen (mit Ausnahme von Query-Strings) so ändern, dass sie exakt mit den Namen in der Datenbanklösung übereinstimmen, einschließlich Groß-/Kleinschreibung.

  In dieser Anweisung:

  ```
  <xsl:value-of select="fmrs:field[@name='LastName']"/>
  ```

  Der Feldverweis LastName in diesem Beispiel muss exakt mit dem Namen des Felds LastName in der Datenbank übereinstimmen.

  Hinweis In XSLT-CWP unterscheiden Feld- und Datenbanknamen, die in Query-Strings verwendet werden, keine Groß-/Kleinschreibung. Weitere Informationen finden Sie unter „Richtlinien für die Verwendung von Query-Befehlen und -Parametern“ auf Seite 96.

- In CDML konnten Sie Felder vergleichen, ohne das Feldattribut anzugeben. Sie könnten z. B. [FMP-If: myfield.eq.10] oder [FMP-If: field:myfield.eq.10] verwenden. Da das Feldattribut in diesem Beispiel nicht im Vergleich eingeschlossen ist, konvertiert der CDML Converter „myfield“ als Stringfolge anstelle eines Feldnamens.

  Nach der Konvertierung wird diese CDML-Anweisung:

  ```
  [FMP-If: myfield.eq.10]
  ```

  in die folgenden XSLT-CWP-Anweisungen umgewandelt:

  ```
  <xsl:choose>
      <xsl:when test="'myfield' = '10'">Ten</xsl:when>
  </xsl:choose>
  ```

  Um diese Art von Problem zu beheben, müssen Sie die Anweisung manuell so ändern, dass die Vergleichsanweisungen im konvertierten Stylesheet den korrekten Feldnamen enthalten. Alternativ können Sie der CDML-Datei an den entsprechenden Stellen field: hinzufügen und die Datei erneut konvertieren.

- Der CDML Converter korrigiert mehrere Instanzen von ungültiger HTML, die von Claris Home Page generiert werden, und nimmt dabei an, dass die Metatags am Beginn der HTML-Seite eingeschlossen sind. Wenn die Metatags entfernt wurden, korrigiert der CDML Converter die ungültige HTML nicht. Claris Home Page oder andere HTML-Dateien könnten weitere Instanzen von ungültiger HTML enthalten, die der CDML Converter nicht korrigieren und korrekt in XHTML konvertieren kann, das strenger strukturiert ist als HTML. Wenn die ungültige HTML nicht korrekt konvertiert wurde, müssen Sie die XHTML im konvertierten Stylesheet korrigieren.

- Der CDML Converter fügt allen Verweisen auf Bilddateien die Dateinamenerweiterung .jpg hinzu, wenn er –img-Aktions-Tags konvertiert. Beispielsweise konvertiert der CDML Converter diese Abfrage:

  /fmpro?–db=products.fp5&–format=format_file.html&–lay=sales&–recid=123&–img

  in diese XSLT-CWP-Abfrage:

  /fmi/xsl/data.jpg?–db=products&–lay=sales&–recid=123

  Wenn die Dateinamenerweiterung .jpg für Ihre Lösung falsch ist, müssen Sie die Erweiterung in Dateinamenverweisen in den konvertierten Stylesheets manuell ändern.

- Der CDML Converter kann keine verschachtelten (eingebetteten) Formulare konvertieren. Wenn Sie Formulare verschachteln, müssen Sie entweder Ihre CDML-Lösung ändern oder die resultierende .xsl-Datei korrigieren.
- Der CDML Converter kann keine CDML-Tags konvertieren, die in ein SGML-Tag eingebettet sind, wie in diesem Beispiel:

```
<input type="text" NAME="StateProvince" SIZE="16" [FMP-If:currentdatabase.eq.Customers.fp5]
VALUE="[FMP-Field:StateProvince]" [/FMP-If]>
    <SELECT NAME="–max">
        <OPTION [FMP-If:currentmax.eq.5] SELECTED[/FMP-If]>5
        <OPTION [FMP-If:currentmax.eq.10] SELECTED[/FMP-If]>10
    </SELECT>
```

  Um Konvertierungsprobleme zu vermeiden, die aus dem vorangehenden Beispiel entstehen, ändern Sie den CDML-Quellcode wie folgt:

```
[FMP-If: currentdatabase.eq.Customers.fp5]
    <input type="text" NAME="StateProvince" SIZE="16" VALUE="[FMP-Field:StateProvince]">
[FMP-Else]
    <input type="text" NAME="StateProvince" SIZE="16" VALUE="">
[/FMP-If]
    <SELECT NAME="–max">
        [FMP-If:currentmax.eq.5]
            <OPTION SELECTED>5 </OPTION>
        [FMP-Else]
            <OPTION>5 </OPTION>
        [FMP-If:currentmax.eq.10]
            <OPTION SELECTED>10 </OPTION>
        [FMP-Else]
            <OPTION>10 </OPTION>
        [/FMP-If]
    </SELECT>
```

- Der CDML Converter entfernt Instanzen des Tags –format in einem Formular und konvertiert seinen Wert in ein Attribut des Formular-Tags. Das Ergebnis im konvertierten Stylesheet enthält einen Eingabeparameter weniger im Formular. Daraus können Probleme entstehen, wenn Ihre CDML-Lösung mithilfe von JavaScript auf Formularfelder verweist.

  Folgendes Formular z. B.:

  ```
  <FORM METHOD="POST" ACTION="FMPro" NAME="checkoutform">
      <INPUT TYPE="hidden" NAME="–db" VALUE="Orders.FP5">
      <INPUT TYPE="hidden" NAME="–format" VALUE="thanks.htm">
      <INPUT TYPE="hidden" NAME="–lay" VALUE="CGI">
      <INPUT TYPE="text" SIZE="50" NAME="Account Number">
  </FORM>
  ```

  wird in die folgenden XSLT-Anweisungen konvertiert:

  ```
  <form method="POST" name="checkoutform"><xsl:attribute name="action">thanks.xsl</xsl:attribute>
      <input type="hidden" name="–db" value="Orders"/>
      <input type="hidden" name="–lay" value="CGI"/>
      <input type="text" size="50" name="Account Number"/>
  </form>
  ```

  Das Problem besteht darin, dass mithilfe von JavaScript sichergestellt wird, dass der Benutzer einen Wert in das Feld „Account Number“ sendet:

  ```
  <P>
  <CENTER>
      <A HREF="javascript:document.checkoutform.elements[3].value == '' ?
      alert('You must fill out the \'Account number\' field!') : document.checkoutform.submit()">
      <IMG SRC="images/continue1.gif" NAME="cont" ALT="Continue"></A>
  </CENTER>
  </P>
  ```

  Nach der Konvertierung dieses Beispiels enthält „checkoutform“ weniger Elemente, da das Tag –format entfernt wurde. Um das Problem in diesem Beispiel zu beheben, müssen Sie das JavaScript manuell so ändern, dass document.checkoutform.elements[2] anstelle von document.checkoutform.elements[3] verwendet wird.

- Die Logik für die Ausdrucksberechnung in CDML unterscheidet sich von der Logik in XSLT. Beispielsweise gibt der folgende CDML-Ausdruck userchoice less than 1 aus, wenn userchoice leer ist:

  ```
  [fmp-if: currentcookie:userchoice .lt. 1 ]
      userchoice less than 1
  [fmp-else]
      userchoice not less than 1
  [/fmp-if]
  ```

  In XSLT gibt derselbe konvertierte Ausdruck userchoice not less than 1 aus, wenn userchoice leer ist:

  ```
  <xsl:choose>
      <xsl:when test="fmxslt:get_cookie('userchoice') &lt; '1'">
          userchoice less than 1
      </xsl:when><xsl:otherwise>
          userchoice not less than 1
  ```

```
        </xsl:otherwise>
    </xsl:choose>
```

Der Grund für diesen Unterschied liegt daran, dass der CDML-Ausdruck einen numerischen Vergleich, der XSLT-Ausdruck aber einen String-Vergleich durchführt. Um in XSLT dasselbe Ergebnis wie der originale CDML-Ausdruck zu liefern, ändern Sie die XSLT-Anweisungen so, dass sie den leeren String berücksichtigen. Beispiel:

```
<xsl:variable name="userchoice" select="fmxslt:get_cookie('userchoice')"/>
<xsl:choose>
        <xsl:when test="string-length($userchoice) = 0 or $userchoice &lt; '1'">
                userchoice less than 1
        </xsl:when><xsl:otherwise>
                userchoice not less than 1
        </xsl:otherwise>
</xsl:choose>
```

Konvertierung von CDML-Ersetzungs-Tags in XSLT-CWP

Dieser Abschnitt beschreibt die Konvertierung von CDML-Ersetzungs-Tags in XSLT-CWP-Anweisungen. In allen Fällen gilt:

- Die konvertierten XSLT-CWP-Anweisungen verwenden dieselbe fmresultset-Grammatik.
- Wenn eine XSLT-CWP-Anweisung einen Wert in einem Attributwert eines SGML-Elements zurückgibt, wird die Form {name} verwendet. Andernfalls wird die Form <xsl:value-of select="name" /> verwendet.
- Wenn ein CDML-Tag einen *Encoding*-Parameter besitzt, wird der konvertierte XSLT-CWP-Wert durch eine der folgenden Erweiterungsfunktionen zur Kodierung übergeben: fmxslt:break_encode(), fmxslt:html_encode () oder fmxslt:url_encode (). Weitere Informationen finden Sie unter „Verwenden der Erweiterungsfunktionen zur Stringmanipulation“ auf Seite 78. Der Kodierungswert Raw gibt nichts an. Wenn die Kodierung auf Raw gesetzt ist, setzt der CDML Converter das Attribut disable-output-escaping der Elemente <xsl:value-of> und <xsl:text> auf yes.
- Innerhalb eines HTML-Dokuments wird die Erweiterungsfunktion fmxslt:html_encode () nicht verwendet.

CDML-Tagname: Client Address (Client-Adresse)

Tag wird durch IP-Adresse des aktuellen Web-Clients ersetzt.

CDML-Syntax: [FMP-ClientAddress]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {$client-ip}
- sonst: <xsl:value-of select="$client-ip"/>

Konvertierungsbeispiel

Original-CDML:	Ihre Adresse lautet: **[FMP-ClientAddress]**
Konvertierte XSLT-CWP:	Ihre Adresse lautet: **<xsl:value-of select="$client-ip"/>**
Konvertiertes Ergebnis:	Ihre Adresse lautet: **192.168.123.101**

CDML-Tagname: Client IP Address (Client-IP-Adresse)

Tag wird durch IP-Adresse des aktuellen Clients ersetzt.

CDML-Syntax: [FMP-ClientIP]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {$client-ip}
- sonst: <xsl:value-of select="$client-ip"/>

Konvertierungsbeispiel

Original-CDML:	Ihre IP-Adresse lautet: **[FMP-ClientIP]**
Konvertierte XSLT-CWP:	Ihre IP-Adresse lautet: **<xsl:value-of select="$client-ip"/>**
Konvertiertes Ergebnis:	Ihre IP-Adresse lautet: **192.168.123.101**

CDML-Tagname: Client Password (Client-Passwort)

Tag wird durch das HTTP-authentifizierte Passwort des Clients ersetzt.

CDML-Syntax: [FMP-ClientPassword]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {$client-password}
- sonst: <xsl:value-of select="$client-password"/>

Konvertierungsbeispiel

Original-CDML:	Ihr Passwort lautet: **[FMP-ClientPassword]**
Konvertierte XSLT-CWP:	Ihr Passwort lautet: **<xsl:value-of select="$client-password"/>**
Konvertiertes Ergebnis:	Ihr Passwort lautet: **my-password**

CDML-Tagname: Client Type (Clienttyp)

Tag wird durch den aktuellen Browsertyp des Clients ersetzt.

CDML-Syntax: [FMP-ClientType]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {fmxslt:get_header('User-Agent')}
- sonst: <xsl:value-of select="fmxslt:get_header('User-Agent')" />

Konvertierungsbeispiel

Original-CDML:	Ihr Browsertyp lautet: **[FMP-ClientType]**
Konvertierte XSLT-CWP:	Ihr Browsertyp lautet: **<xsl:value-of select="fmxslt:get_header('User-Agent')" />**
Konvertiertes Ergebnis:	Ihr Browsertyp lautet: **Mozilla/3.01 (Macintosh; I; PPC)**

CDML-Tagname: Client User Name (Client-Benutzername)

Tag wird durch den Benutzernamen des Clients aus der HTTP-Authentifizierung ersetzt.

CDML-Syntax: [FMP-ClientUserName]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {$client-user-name}
- sonst: <xsl:value-of select="$client-user-name"/>

Konvertierungsbeispiel

Original-CDML:	Ihr Name lautet: **[FMP-ClientUserName]**
Konvertierte XSLT-CWP:	Ihr Name lautet: **<xsl:value-of select="$client-user-name"/>**
Konvertiertes Ergebnis:	Ihr Name lautet: **my-user-name**

CDML-Tagname: Content MIME Type (MIME-Inhaltstyp)

Tag wird in der HTML mit nichts ersetzt und der an den Browser zurückgegebene MIME-Typ wird geändert.

CDML-Syntax: [FMP-ContentMimeType: MimeType]

XSLT-CWP-Konvertierung:

- <xsl:variable name="header1" select="fmxslt:set_header('Content-Type', 'MimeType')" />
- wenn innerhalb des SGML-Element-Attributwerts: Einfügung außerhalb/nach dem Element

Hinweis Da nach der Konvertierung die Erweiterungsfunktion fmxslt:set_header() verwendet wird, muss die Inhaltspufferung aktiviert werden. Dafür fügt der CDML Converter automatisch die Verarbeitungsanweisung <?xslt-cwp-buffer?> ein. Weitere Informationen finden Sie unter „Verwenden von Inhaltspufferung“ auf Seite 70.

Konvertierungsbeispiel

Original-CDML:	Ein Beispiel für **[FMP-ContentMIMEType: text/plain]** Text.
Konvertierte XSLT-CWP:	Ein Beispiel für **<xsl:variable name="header1" select="fmxslt:set_header('Content-Type', 'text/plain')" />** Text.
Konvertiertes Ergebnis:	Ein Beispiel für Text. (Der Header der HTTP-Antwort enthält auch **Content-Type: text/plain** .)

CDML-Tagname: Cookie

Tag wird durch den aktuellen Wert des angegebenen Cookies ersetzt.

CDML-Syntax: [FMP-Cookie: *CookieName*, *Kodierung*], wobei *Kodierung* 'Raw' (Standard) oder 'URL' ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {fmxslt:get_cookie('*CookieName*')}
- sonst: <xsl:value-of select="fmxslt:get_cookie('*CookieName*')" />

Konvertierungsbeispiel

Original-CDML:	Willkommen zurück! Die neuesten Produkte in **[FMP-Cookie: ColorChoice]** sind nachstehend aufgelistet.
Konvertierte XSLT-CWP:	Willkommen zurück! Die neuesten Produkte in **<xsl:value-of select="fmxslt:get_cookie('ColorChoice')" />** sind nachstehend aufgelistet.
Konvertiertes Ergebnis:	Willkommen zurück! Die neuesten Produkte in **Grün** sind nachstehend aufgelistet.

CDML-Tagname: Current Action (Aktuelle Aktion)

Tag wird durch den Namen der aktuellen Aktion ersetzt, z. B.: find, findall, new, edit, delete, view oder dup. Der zurückgegebene Name enthält keinen Gedankenstrich (–).

CDML-Syntax: [FMP-CurrentAction: *Kodierung*], wobei *Kodierung* 'HTML' (Standard) oder 'Display' ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {$current-action}
- sonst: <xsl:value-of select="$current-action" />
- wenn *Format* 'Display' ist: <!—CDML2XSLT-WARNUNG: [FMP-CurrentAction] Die Kodierung 'Display' wird nicht unterstützt. -->
- falls im Dokument verwendet, wird die Variable $current-action mithilfe der benannten Vorlage des Stylesheets cdml2xsl_utilities.xsl auf der obersten Ebene erstellt.

Konvertierungsbeispiel

Original-CDML:	Letzte Aktion war: **[FMP-CurrentAction]**
Konvertierte XSLT-CWP:	Letzte Aktion war: **<xsl:value-of select="$current-action" />**
Konvertiertes Ergebnis:	Letzte Aktion war: **view**

CDML-Tagname: Current Database (Aktuelle Datenbank)

Tag wird durch den Namen der gerade verarbeiteten Datenbank ersetzt.

CDML-Syntax: [FMP-CurrentDatabase: *Kodierung*], wobei *Kodierung* 'Raw', 'URL' oder 'HTML' (Standard) ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {fmrs:fmresultset/fmrs:datasource/@database}
- sonst: <xsl:value-of select="fmrs:fmresultset/fmrs:datasource/@database"/>

Konvertierungsbeispiel

Original-CDML:	Sie sehen die Datenbank **[FMP-CurrentDatabase]** an.
Konvertierte XSLT-CWP:	Sie sehen die Datenbank **<xsl:value-of select="fmrs:fmresultset/fmrs:datasource/@database" />** an.
Konvertiertes Ergebnis:	Sie sehen die Datenbank **Contact** an.

CDML-Tagname: Current Date (Aktuelles Datum)

Tag wird durch das aktuelle Datum ersetzt.

CDML-Syntax: [FMP-CurrentDate: *Format*], wobei *Format* 'Short' (Standard) oder 'Long' ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts, 'Short': {fmxslt:get_date('short')}:
- oder 'Long': {fmxslt:get_date('long')}
- sonst 'Short': <xsl:value-of select="fmxslt:get_date('short')" />
- oder 'Long': <xsl:value-of select="fmxslt:get_date('long')" />
- wenn *Format* gleich 'Abbrev' ist: <!-- CDML CONVERTER-WARNUNG: Das abgekürzte Datumsformat wird nicht mehr unterstützt – verwenden Sie fmxslt:get_datetime() für zusätzliche Formatierung.>

Konvertierungsbeispiel

Original-CDML:	Aktuelles Datum: **[FMP-CurrentDate]**
Konvertierte XSLT-CWP:	Aktuelles Datum: **<xsl:value-of select="fmxslt:get_date()" />**
Konvertiertes Ergebnis:	Aktuelles Datum: **2.2.04**

CDML-Tagname: Current Day (Aktueller Tag)

Tag wird durch den Namen des aktuellen Wochentags ersetzt.

CDML-Syntax: [FMP-CurrentDay: *Format*], wobei *Format* 'Short' (Standard) oder 'Long' ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts, 'Short': {fmxslt:get_day('short')}
- oder 'Long': {fmxslt:get_day('long')}
- sonst 'Short': <xsl:value-of select="fmxslt:get_day('short')" />
- oder 'Long': <xsl:value-of select="fmxslt:get_day('long')" />

Konvertierungsbeispiel

Original-CDML:	Aktueller Wochentag: **[FMP-CurrentDay: Long]**
Konvertierte XSLT-CWP:	Aktueller Wochentag: **<xsl:value-of select="fmxslt:get_day('long')" />**
Konvertiertes Ergebnis:	Aktueller Wochentag: **Monday**

CDML-Tagname: Current Error (Aktueller Fehler)

Tag wird durch die FileMaker-Fehlernummer aus der aktuellen Aktion ersetzt.

CDML-Syntax: [FMP-CurrentError]

XSLT-CWP-Konvertierung:

wenn innerhalb des SGML-Element-Attributwerts: {fmrs:fmresultset:/fmrs:error/@code}

sonst: <xsl:value-of select="fmrs:fmresultset:/fmrs:error/@code" />

Konvertierungsbeispiel

Original-CDML:	Vorherige Operation fehlgeschlagen mit Fehlernummer **[FMP-CurrentError]**.
Konvertierte XSLT-CWP:	Vorherige Operation fehlgeschlagen mit Fehlernummer **<xsl:value-of select="fmrs:fmresultset:/fmrs:error/@code" />**.
Konvertiertes Ergebnis:	Vorherige Operation fehlgeschlagen mit Fehlernummer **500**.

CDML-Tagname: Current Find (Aktuelle Suche)

Wiederholt die HTML zwischen [FMP-CurrentFind] und [/FMP-CurrentFind] für jedes Suchkriterium, das Teil der Abfrage war, die diese Seite erzeugt hat.

CDML-Syntax: [FMP-CurrentFind]...[/FMP-CurrentFind]

XSLT-CWP-Konvertierung:

- <xsl:for-each select="$current-find">...</xsl:for-each>
- wenn in einem SGML-Tag/Attribut, <!—CDML Converter-FEHLER: [FMP-CurrentFind] war nicht richtig verschachtelt -->
- falls im Dokument verwendet, wird die Variable $current-find mithilfe der benannten Vorlage des Stylesheets cdml2xsl_utilities.xsl auf der obersten Ebene erstellt.

Konvertierungsbeispiel

Original-CDML:	Aktuelle Suchabfrage: **[FMP-CurrentFind]** Field: [FMP-FindFieldItem], Op: [FMP-FindOpItem], Value: [FMP-FindValueItem] [/FMP-CurrentFind]
Konvertierte XSLT-CWP:	Aktuelle Suchabfrage: **<xsl:for-each select="$current-find/find-field">** Field: <xsl:value-of select="@name" />, Op: <xsl:value-of select="@long-operator" />, Value: <xsl:value-of select="text()" /> </xsl:for-each>
Konvertiertes Ergebnis:	Aktuelle Suchabfrage: Field: First Name, Op: begins with, Value: John Field: Last Name, Op: equals, Value: Doe

CDML-Tagname: Current Format File (Aktuelle Formatdatei)

Tag wird durch den Namen der aktuellen Formatdatei ersetzt.

CDML-Syntax: [FMP-CurrentFormat: *Kodierung*], wobei *Kodierung* 'Raw', 'URL' oder 'HTML' (Standard) ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {$request-query/@action}
- sonst: <xsl:value-of select="$request-query/@action"/>

Konvertierungsbeispiel

Original-CDML:	Diese Seite wurde formatiert mit Datei: **[FMP-CurrentFormat]**.
Konvertierte XSLT-CWP:	Diese Seite wurde formatiert mit Datei: **<xsl:value-of select="$request-query/@action"/>**.
Konvertiertes Ergebnis:	Diese Seite wurde formatiert mit Datei: **Detail.xsl**.

CDML-Tagname: Current Found Count (Aktuell aufgerufene Datensätze)

Tag wird durch die Anzahl der Datensätze in der aktuellen Ergebnismenge ersetzt.

CDML-Syntax: [FMP-CurrentFoundCount]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {fmrs:fmresultset/fmrs:resultset/@count}
- sonst: <xsl:value-of select="fmrs:fmresultset/fmrs:resultset/@count" />

Konvertierungsbeispiel

Original-CDML:	Datensätze in Ergebnismenge: **[FMP-CurrentFoundCount]**
Konvertierte XSLT-CWP:	Datensätze in Ergebnismenge: **<xsl:value-of select="fmrs:fmresultset/fmrs:resultset/@count" />**
Konvertiertes Ergebnis:	Datensätze in Ergebnismenge: **12**

CDML-Tagname: Current Layout (Aktuelles Layout)

Tag wird durch den Namen des Layouts ersetzt, mit dem die Seite verarbeitet wird.

CDML-Syntax: [FMP-CurrentLayout: *Kodierung*], wobei *Kodierung* 'Raw', 'URL' oder 'HTML' (Standard) ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {fmrs:fmresultset/fmrs:datasource/@layout}
- sonst: <xsl:value-of select="fmrs:fmresultset/fmrs:datasource/@layout" />

Konvertierungsbeispiel

Original-CDML:	Diese Seite wurde formatiert mit dem Layout: **[FMP-CurrentLayout]**.
Konvertierte XSLT-CWP:	Diese Seite wurde formatiert mit dem Layout: **<xsl:value-of select="fmrs:fmresultset/fmrs:datasource/@layout" />** layout.
Konvertiertes Ergebnis:	Diese Seite wurde formatiert mit dem Layout: **Detail**.

CDML-Tagname: Current Logical Operator (Aktueller logischer Operator)

Tag wird durch den logischen Operator ersetzt, der für die aktuelle Suche benutzt wird.

CDML-Syntax: [FMP-CurrentLOP]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {$current-lop}
- sonst: <xsl:value-of select="$current-lop" />
- falls im Dokument verwendet, wird die Variable $current-lop mithilfe der benannten Vorlage des Stylesheets cdml2xsl_utilities.xsl auf der obersten Ebene erstellt.

Konvertierungsbeispiel

Original-CDML:	Sie haben gerade eine "**[FMP-CurrentLOP]**"-Suche durchgeführt.
Konvertierte XSLT-CWP:	Sie haben gerade eine "**<xsl:value-of select="$current-lop" />**"-Suche durchgeführt.
Konvertiertes Ergebnis:	Sie haben gerade eine "**or**"-Suche durchgeführt.

CDML-Tagname: Current Max (Aktuelles Maximum)

Tag wird durch das angegebene Maximum an Datensätzen ersetzt.

CDML-Syntax: [FMP-CurrentMax]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {$current-max}
- sonst: <xsl:value-of select="$current-max" />
- falls im Dokument verwendet, wird die Variable $current-max mithilfe der benannten Vorlage des Stylesheets cdml2xsl_utilities.xsl auf der obersten Ebene erstellt.

Konvertierungsbeispiel

Original-CDML:	Klicken Sie auf die Schaltfläche, um die nächsten **[FMP-CurrentMax]**-Datensätze zu sehen.
Konvertierte XSLT-CWP:	Klicken Sie auf die Schaltfläche, um die nächsten **<xsl:value-of select="$current-max" />**-Datensätze zu sehen.
Konvertiertes Ergebnis:	Klicken Sie auf die Schaltfläche, um die nächsten **10** Datensätze zu sehen.

CDML-Tagname: Current Modification ID (Aktuelle Änderungs-ID)

Tag wird durch die Änderungs-ID des aktuellen Datensatzes ersetzt, der bearbeitet wird.

CDML-Syntax: [FMP-CurrentModID]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts:
  - wenn aktueller Kontext ein Datensatz ist: {@mod-id}
  - sonst: {/fmrs:fmresultset/fmrs:resultset/fmrs:record[1]/@mod-id}
- sonst:
  - wenn aktueller Kontext ein Datensatz ist: <xsl:value-of select="@mod-id" />
  - sonst: <xsl:value-of select="/fmrs:fmresultset/fmrs:resultset/fmrs:record[1]/@mod-id" />

Konvertierungsbeispiel

Original-CDML:

```
<FORM ACTION="FMPro" METHOD="POST">
  <INPUT TYPE="HIDDEN" NAME="–DB" VALUE="contacts.fp5">
  <INPUT TYPE="HIDDEN" NAME="–Format" VALUE="results.htm">
  <INPUT TYPE="HIDDEN" NAME="–RecID" VALUE="[FMP-CurrentRecID]">
  <INPUT TYPE="HIDDEN" NAME="–ModID" VALUE="[FMP-CurrentModID]">
  <INPUT TYPE="TEXT" NAME="Country">
  <INPUT TYPE="SUBMIT" NAME="–Edit" VALUE="Edit This Record">
</FORM>
```

Konvertierungsbeispiel

Konvertierte XSLT-CWP:	<form action="/fmi/xsl/results.xsl" method="POST"> <input type="HIDDEN" name="–DB" value="contacts"></input> <input type="HIDDEN" name="–grammar" value="fmresultset"></input> <input type="HIDDEN" name="–RecID" value="{@record-id}"></input> <input type="HIDDEN" name="–ModID" value="**{@mod-id}**"></input> <input type="TEXT" name="Country"></input> <input type="SUBMIT" name="–Edit" value="Edit This Record"></input> </form>
Konvertiertes Ergebnis:	<form action="/fmi/xsl/results.xsl" method="POST"> <input type="HIDDEN" name="–DB" value="contacts"> <input type="HIDDEN" name="–grammar" value="fmresultset"> <input type="HIDDEN" name="–RecID" value="1032"> <input type="HIDDEN" name="–ModID" value="**3**"> <input type="TEXT" name="Country"> <input type="SUBMIT" name="–Edit" value="Edit This Record"> </form>

CDML-Tagname: Current Portal Number (Aktuelle Ausschnittnummer)

Tag wird durch die Zeilennummer des aktuellen Ausschnitts ersetzt, der gerade verarbeitet wird. Diese befindet sich immer in einer [FMP-Portal]-Schleife.

CDML-Syntax: [FMP-CurrentPortalRowNumber]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {position()}
- sonst: <xsl:value-of select="position()"/>
- wenn außerhalb einer [FMP-Portal]-Schleife: <!—CDML Converter-FEHLER: [FMP-CurrentPortalRowNumber] außerhalb von [FMP-Portal] -->

Konvertierungsbeispiel

Original-CDML:	[FMP-Portal: lineitems] **[FMP-CurrentPortalRowNumber]**: [FMP-Field: lineitems::name] [/FMP-Portal]
Konvertierte XSLT-CWP:	<xsl:for-each select="fmrs:relatedset[@table='lineitems']/fmrs:record"> **<xsl:value-of select="position()"/>:** <xsl:value-of select="fmrs:field[@name = 'name']" /> </xsl:for-each>
Konvertiertes Ergebnis:	**1**: Red

CDML-Tagname: Current Record Count (Aktuelle Datensatzanzahl)

Tag wird durch die Gesamtanzahl der Datensätze in der Datenbank ersetzt.

CDML-Syntax: [FMP-CurrentRecordCount]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {fmrs:fmresultset/fmrs:datasource/@total-count}
- sonst: <xsl:value-of select="fmrs:fmresultset/fmrs:datasource/@total-count"/>

Konvertierungsbeispiel

Original-CDML:	Gesamtanzahl der Datensätze in der Datenbank: **[FMP-CurrentRecordCount]**
Konvertierte XSLT-CWP:	Gesamtanzahl der Datensätze in der Datenbank: **<xsl:value-of select="fmrs:fmresultset/fmrs:datasource/@total-count"/>**
Konvertiertes Ergebnis:	Gesamtanzahl der Datensätze in der Datenbank: **1123**

CDML-Tagname: Current Record ID (Aktuelle Datensatz-ID)

Tag wird durch die Datensatz-ID ersetzt.

CDML-Syntax: [FMP-CurrentRecID]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts:
  - wenn aktueller Kontext ein Datensatz ist: {@record-id}
  - sonst: {/fmrs:fmresultset/fmrs:resultset/fmrs:record[1]/@record-id}
- sonst:
  - wenn aktueller Kontext ein Datensatz ist: <xsl:value-of select="@record-id" />
  - sonst: <xsl:value-of select="/fmrs:fmresultset/fmrs:resultset/fmrs:record[1]/@record-id" />

Konvertierungsbeispiel

Original-CDML:	<form action="FMPro" method="post"> <input type="hidden" name="–DB" value="name.fp5"> <input type="hidden" name="–Format" value="results.htm"> <input type="hidden" name="–RecID" value="**[FMP-CurrentRecID]**"> <input type="submit" name="–Delete" value="Delete This Record"> </form>
Konvertierte XSLT-CWP:	<form action="/fmi/xsl/results.xsl" method="post"> <input type="hidden" name="–DB" value="name"></input> <input type="hidden" name="–grammar" value="fmresultset"></input> <input type="hidden" name="–RecID" value="**{@record-id}**"></input> <input type="submit" name="–Delete" value="Delete This Record"></input> </form>

Konvertierungsbeispiel

Konvertiertes Ergebnis:	<form action="/fmi/xsl/results.xsl" method="post"> <input type="hidden" name="–DB" value="name"> <input type="hidden" name="–grammar" value="fmresultset"> <input type="hidden" name="–RecID" value="**1023**"> <input type="submit" name="–Delete" value="Delete This Record"> </form>

CDML-Tagname: Current Record Number (Aktuelle Datensatznummer)

Tag wird durch die Position des Datensatzes in der aktuellen Ergebnismenge ersetzt.

CDML-Syntax: [FMP-CurrentRecordNumber]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {position()}
- sonst: <xsl:value-of select="position() + $current-skip" />
- wenn außerhalb einer [FMP-Portal]-Schleife: <!—CDML Converter-FEHLER: [FMP-CurrentRecordNumber] außerhalb von [FMP-Record] -->
- falls im Dokument verwendet, wird die Variable $current-skip mithilfe der benannten Vorlage des Stylesheets cdml2xsl_utilities.xsl auf der obersten Ebene erstellt.

Konvertierungsbeispiel

Original-CDML:	Aktueller Datensatz in Ergebnismenge: **[FMP-CurrentRecordNumber]**
Konvertierte XSLT-CWP:	Aktueller Datensatz in Ergebnismenge: **<xsl:value-of select="position() + $current-skip" />**
Konvertiertes Ergebnis:	Aktueller Datensatz in Ergebnismenge: **3**

CDML-Tagname: Current Repeat Number (Aktuelle Wiederholungsnummer)

Tag wird durch die Wiederholung ersetzt, die gerade verarbeitet wird. Diese befindet sich immer in einer [FMP-Repeating][/FMP-Repeating]-Schleife.

CDML-Syntax: [FMP-CurrentRepeatNumber]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {position()}
- sonst: <xsl:value-of select="position()"/>
- wenn außerhalb einer [FMP-Repeating]-Schleife: <!—CDML Converter-FEHLER: [FMP-CurrentRepeatNumber] außerhalb von [FMP-Repeating] -->

Konvertierungsbeispiel

Original-CDML:	[FMP-Repeating: extensions] **[FMP-CurrentRepeatNumber]**: [FMP-RepeatingItem] [/FMP-Repeating]
Konvertierte XSLT-CWP:	<xsl:for-each select="fmrs:field[@name = 'extensions']/fmrs:data"> **<xsl:value-of select="position()" />**: <xsl:value-of select="." /> </xsl:for-each>
Konvertiertes Ergebnis:	**3**: Green

CDML-Tagname: Current Skip Setting (Aktuelle Einstellung für Überspringen)

Tag wird durch die Anzahl der Datensätze ersetzt, die ab dem Beginn der Ergebnismenge übersprungen werden.

CDML-Syntax: [FMP-CurrentSkip]

XSLT-CWP-Konvertierung: <xsl:call-template name="$get-current-skip" />

- wenn innerhalb des SGML-Element-Attributwerts: {$current-skip}
- sonst: <xsl:value-of select="$current-skip"/>
- falls im Dokument verwendet, wird die Variable $current-skip mithilfe der benannten Vorlage des Stylesheets cdml2xsl_utilities.xsl auf der obersten Ebene erstellt.

Konvertierungsbeispiel

Original-CDML:	Übersprungene Datensätze: **[FMP-CurrentSkip]**
Konvertierte XSLT-CWP:	Übersprungene Datensätze: **<xsl:value-of select="$current-skip"/>**
Konvertiertes Ergebnis:	Übersprungene Datensätze: **10**

CDML-Tagname: Current Sort Order (Aktuelle Sortierfolge)

Wiederholt die HTML zwischen [FMP-CurrentSort] und [/FMP-CurrentSort] für jedes Sortierkriterium, das Teil der Abfrage war, die diese Seite erzeugt hat.

CDML-Syntax: [FMP-CurrentSort]...[/FMP-CurrentSort]

XSLT-CWP-Konvertierung:

- <xsl:for-each select="$current-sort/sort-field">...</xsl:for-each>
- wenn in einem SGML-Tag/Attribut, <!—CDML Converter-FEHLER: [FMP-CurrentSort] war nicht richtig verschachtelt -->
- falls im Dokument verwendet, wird die Variable $current-sort mithilfe der benannten Vorlage des Stylesheets cdml2xsl_utilities.xsl auf der obersten Ebene erstellt.

Konvertierungsbeispiel

Original-CDML:	Aktuelle Sortierfolge: **[FMP-CurrentSort]** Field: [FMP-SortFieldItem], Order: [FMP-SortOrderItem] [/FMP-CurrentSort]
Konvertierte XSLT-CWP:	Aktuelle Sortierfolge: **<xsl:for-each select="$current-sort/sort-field">** Field: <xsl:value-of select="@name" />, Order: <xsl:value-of select="@order" /> </xsl:for-each>
Konvertiertes Ergebnis:	Aktuelle Sortierfolge: Field: First Name, Order: descend Field: Last Name, Order: descend

CDML-Tagname: Current Time (Aktuelle Uhrzeit)

Tag wird durch die aktuelle Uhrzeit ersetzt.

CDML-Syntax: [FMP-CurrentTime: *Format*], wobei *Format* 'Short' (Standard) oder 'Long' ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts 'Short': {fmxslt:get_time('short')}
- oder 'Long': {fmxslt:get_time('long')}
- sonst 'Short': <xsl:value-of select="fmxslt:get_time('short)" />
- oder 'Long': <xsl:value-of select="fmxslt:get_time('long)" />

Konvertierungsbeispiel

Original-CDML:	Aktuelle Uhrzeit: **[FMP-CurrentTime: Short]**
Konvertierte XSLT-CWP:	Aktuelle Uhrzeit: **<xsl:value-of select="fmxslt:get_time('short')" />**
Konvertiertes Ergebnis:	Aktuelle Uhrzeit: **10:12:00**

CDML-Tagname: Current Token (Aktueller Token)

Tag wird durch den Namen des –Token-Variablen-Tags ersetzt, mit dem die aktuelle Seite erstellt wurde.

CDML-Syntax: [FMP-CurrentToken: Nummer, Kodierung], wobei *Nummer* '0'...'9' (optional) und *kodierung* 'Raw', 'URL' oder 'HTML' (Standard) sind.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {$request-query/fmq:query/fmq:parameter[@NAME='–token']}
- sonst: <xsl:value-of select="$request-query/fmq:query/fmq:parameter[@NAME='–token']" />
- wenn der Token eine *Nummer* hat, lautet der Query-Parameter '–token.*Nummer*'. Beispiel: '–token.5

Konvertierungsbeispiel

Original-CDML:	Der Wert von token.5 lautet **[FMP-CurrentToken: 5, HTML]**
Konvertierte XSLT-CWP:	Der Wert von token.5 lautet **<xsl:value-of select="$request-query/fmq:query/fmq:parameter[@NAME='-token.5']" />**
Konvertiertes Ergebnis:	Der Wert von token.5 lautet **MyValue**

CDML-Tagname: Else (Sonst)

Tag wird durch die angegebenen Daten ersetzt, wenn der vorherige boolesche Ausdruck [FMP-If] falsch ist.

CDML-Syntax: [FMP-If: BooleanExpression1]...[FMP-Else]...[/FMP-If]

XSLT-CWP-Konvertierung:

- <xsl:choose><xsl:when test="BooleanExpression1">...</xsl:when><xsl:otherwise>...</xsl:otherwise></xsl:choose>
- wenn in einem SGML-Tag/Attribut, <!—CDML Converter-FEHLER: [FMP-Else] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	FMP-If: (Field:country.Eq.us).or.(Field:country.Eq.usa)] United States of America **[FMP-Else]** Other country [/FMP-If]
Konvertierte XSLT-CWP:	<xsl:choose> <xsl:when test="'fmrs:field[@name = 'country']/fmrs:data[1] = 'us' or fmrs:field[@name = 'country']/fmrs:data[1] = 'usa'">United States of America**</xsl:when>** **<xsl:otherwise>**Other country**</xsl:otherwise>** </xsl:choose>
Konvertiertes Ergebnis:	Other country

CDML-Tagname: Else If (Sonst, wenn)

Tag wird durch die angegebenen Daten ersetzt, wenn BooleanExpression wahr ist und der vorherige boolesche Ausdruck [FMP-If] falsch ist.

CDML-Syntax: [FMP-If: BooleanExpression1]...[FMP-ElseIf BooleanExpression2]...[/FMP-If]

XSLT-CWP-Konvertierung:

- <xsl:choose><xsl:when test="BooleanExpression1">...</xsl:when><xsl:when test="BooleanExpression2">...</xsl:when></xsl:choose>
- BooleanExpression2 wird zu XPath konvertiert.
- wenn in einem SGML-Tag/Attribut, <!—CDML Converter-FEHLER: [FMP-ElseIf] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	[FMP-If: (Field:country.Eq.us).or.(Field:country.Eq.usa)] United States of America **[FMP-ElseIf: Field:country .Eq. Italy]** Italy [/FMP-If]
Konvertierte XSLT-CWP:	<xsl:choose> <xsl:when test="'fmrs:field[@name = 'country']/fmrs:data[1] = 'us' or fmrs:field[@name = 'country']/fmrs:data[1] = 'usa'">United States of America**</xsl:when>** **<xsl:when test="fmrs:field[@name = 'country']/fmrs:data[1] = 'Italy'">**Italy</xsl:when> </xsl:choose>
Konvertiertes Ergebnis:	Italy

CDML-Tagname: Feld

Tag wird durch den aktuellen Inhalt des angegebenen Felds ersetzt.

CDML-Syntax: [FMP-Field: FeldName, *Kodierung*], wobei *Kodierung* 'Raw', 'URL', 'HTML' (Standard) oder 'Break' ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts:
  - wenn aktueller Kontext ein Datensatz ist: {fmrs:field[@name = 'FieldName']/fmrs:data[1]}
  - sonst: {/fmrs:fmresultset/fmrs:resultset/fmrs:record[1]/fmrs:field[@name = '*FeldName*']/fmrs:data[1]}
- sonst:
  - wenn aktueller Kontext ein Datensatz ist: <xsl:value-of select="fmrs:field[@name = '*FieldName*']/fmrs:data[1]" />
  - sonst: <xsl:value-of select="/fmrs:fmresultset/fmrs:resultset/fmrs:record[1]/fmrs:field[@name = '*FieldName*']/fmrs:data[1]" />

Konvertierungsbeispiel

Original-CDML:	Vorname: **[FMP-Field: First Name]**
Konvertierte XSLT-CWP:	Vorname: **<xsl:value-of select="fmrs:field[@name = 'First Name']/fmrs:data[1]" />**
Konvertiertes Ergebnis:	Vorname: **Johann**

CDML-Tagname: Field Name (Feldname)

Tag wird durch den Namen des aktuellen Felds ersetzt. Dieser befindet sich immer in einer [FMP-LayoutFields]-Schleife.

CDML-Syntax: [FMP-FieldName: *Kodierung*], wobei *kodierung* 'Raw', 'URL' oder 'HTML' (Standard) ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {@name}
- sonst: <xsl:value-of select="@name" />
- wenn außerhalb einer [FMP-LayoutFields]-Schleife: <!—CDML Converter-FEHLER: [FMP-FieldName] außerhalb von [FMP-LayoutFields] -->

Konvertierungsbeispiel

Original-CDML:	<select name="-SortField"> <option value="">-None- [FMP-LayoutFields] <option>**[FMP-FieldName: Raw]** [/FMP-LayoutFields] </select>
Konvertierte XSLT-CWP:	<select name="-SortField"> <option value="" />-None- <xsl:for-each select="fmrs:fmresultset/fmrs:metadata/fmrs:field-definition" /> <option>**<xsl:value-of select="@name"/>** </option> </xsl:for-each> </select>
Konvertiertes Ergebnis:	<select name="-SortField"> <option value="">-None-</option> <option **First Name**</option> <option **Last Name**</option> <option **Employee Number**</option> </select>

CDML-Tagname: Find Field Item (Suchfeld)

Tag wird durch den Feldnamen ersetzt, der Teil der Suchabfrage für die Erstellung dieser Seite war. Dieser befindet sich immer in einer [FMP-CurrentFind]-Schleife.

CDML-Syntax: [FMP-FindFieldItem: *Kodierung*], wobei *Kodierung* 'Raw', 'URL' oder 'HTML' (Standard) ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {@name}
- sonst: <xsl:value-of select="@name" />
- falls im Dokument verwendet, wird die Variable $current-find mithilfe der benannten Vorlage des Stylesheets cdml2xsl_utilities.xsl auf der obersten Ebene erstellt.
- wenn außerhalb einer [FMP-CurrentFind]-Schleife: <!—CDML Converter-FEHLER: [FMP-FindFieldItem] außerhalb von [FMP-CurrentFind] -->

Konvertierungsbeispiel

Original-CDML:	Aktuelle Suchabfrage: [FMP-CurrentFind] Feld: **[FMP-FindFieldItem]**, Op: [FMP-FindOpItem], Value: [FMP-FindValueItem] [/FMP-CurrentFind]
Konvertierte XSLT-CWP:	Aktuelle Suchabfrage: <xsl:for-each select="xalan:nodeset($current-find)/find-field"> Field: **<xsl:value-of select="@name" />**, Op: <xsl:value-of select="@long-operator" />, Value: <xsl:value-of select="text()" /> </xsl:for-each>
Konvertiertes Ergebnis:	Aktuelle Suchabfrage: Field: **First Name**, Op: begins with, Wert: John Field: **Last Name**, Op: equals, Wert: Doe

CDML-Tagname: Find Operator Item (Suchoperator)

Tag wird durch den Suchoperator ersetzt, der Teil der Suchabfrage für die Erstellung dieser Seite war. Dieser befindet sich immer in einer [FMP-CurrentFind]-Schleife.

CDML-Syntax: [FMP-FindOpItem: *Format*], wobei *Format* 'Short', 'Long' (Standard) oder 'Display' ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {@short-operator} oder {@long-operator}
- sonst: <xsl:value-of select="@short-operator" /> oder <xsl:value-of select="@long-operator" />
- wenn *Format* 'Display' ist, wird 'Long' verwendet und <!—CDML2XSLT-WARNUNG: [FMP-SortOrderItem] Format 'Display' wird von XSLT-CWP nicht unterstützt. -->
- falls im Dokument verwendet, wird die Variable $current-find mithilfe der benannten Vorlage des Stylesheets cdml2xsl_utilities.xsl auf der obersten Ebene erstellt.
- wenn außerhalb einer [FMP-CurrentFind]-Schleife: <!—CDML Converter-FEHLER: [FMP-FindOpItem] außerhalb von [FMP-CurrentFind] -->

Konvertierungsbeispiel

Original-CDML:	Aktuelle Suchabfrage: [FMP-CurrentFind] Field: [FMP-FindFieldItem], Op: **[FMP-FindOpItem]**, Value: [FMP-FindValueItem] [/FMP-CurrentFind]
Konvertierte XSLT-CWP:	Aktuelle Suchabfrage: <xsl:for-each select="xalan:nodeset($current-find)/find-field"> Field: <xsl:value-of select="@name" />, Op: **<xsl:value-of select="@long-operator" />**, Value: <xsl:value-of select="text()" /> </xsl:for-each>
Konvertiertes Ergebnis:	Aktuelle Suchabfrage: Field: First Name, Op: **begins with**, Value: John Field: Last Name, Op: **equals**, Value: Doe

CDML-Tagname: Find Value Item (Suchwert)

Tag wird durch den Suchwert ersetzt, der Teil der Suchabfrage für die Erstellung dieser Seite war. Dieser befindet sich immer in einer [FMP-CurrentFind]-Schleife.

CDML-Syntax: [FMP-FindValueItem: *Kodierung*], wobei *Kodierung* 'Raw', 'URL' oder 'HTML' (Standard) ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {text()}
- sonst: <xsl:value-of select="text()" />
- falls im Dokument verwendet, wird die Variable $current-find mithilfe der benannten Vorlage des Stylesheets cdml2xsl_utilities.xsl auf der obersten Ebene erstellt.
- wenn außerhalb einer [FMP-CurrentFind]-Schleife: <!—CDML Converter-FEHLER: [FMP-FindValueItem] außerhalb von [FMP-CurrentFind] -->

Konvertierungsbeispiel

Original-CDML:	Aktuelle Suchabfrage: [FMP-CurrentFind] Field: [FMP-FindFieldItem], Op: [FMP-FindOpItem], Value: **[FMP-FindValueItem]** [/FMP-CurrentFind]
Konvertierte XSLT-CWP:	Aktuelle Suchabfrage: <xsl:for-each select="xalan:nodeset($current-find)/find-field"> Field: <xsl:value-of select="@name" />, Op: <xsl:value-of select="@long-operator" />, Value: **<xsl:value-of select="text()" />** </xsl:for-each>
Konvertiertes Ergebnis:	Aktuelle Suchabfrage: Field: First Name, Op: begins with, Wert: **John** Field: Last Name, Op: equals, Wert: **Doe**

CDML-Tagname: Header

Der HTTP-Header der an den Browser gesendeten Seite wird durch den Text zwischen den Tags [FMP-Header] und [/FMP-Header] ersetzt. Text, der sich zwischen den Tags befindet, wird nicht auf dem HTML-Teil der Seite angezeigt.

CDML-Syntax: [FMP-Header]...[/FMP-Header]

XSLT-CWP-Konvertierung:

- wenn sich der Statuscode im Header-Inhalt befindet: <xsl:variable name="header-status-code1" select="set_status_code('<Statuscode aus Inhalten>')" />
- für jedes Name-Wert-Paar im Header-Inhalt: <xsl:variable name="header-param*NUMBER*" select="fmxslt:set_header('<ParamName>', '<ParamWert>')" />
- wenn in einem SGML-Tag/Attribut, <!—CDML Converter-FEHLER: [FMP-Header] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	Header-Code ist hier: **[FMP-Header]** **HTTP/1.0 302 Moved Temporary** **Location: http://www.FileMaker.com** [/FMP-Header]
Konvertierte XSLT-CWP:	Header-Code ist hier: **<xsl:variable name="header-status-code1" select="set_status_code(302)" />** **<xsl:variable name="header-param1" select="fmxslt:set_header('location', 'http://www.FileMaker.com')" />**
Konvertiertes Ergebnis:	Header-Code ist hier: (Die HTTP-Antwort enthält auch **Location: http://www.filemaker.com** im Header.)

CDML-Tagname: Wenn

Zusammen mit den Tags [FMP-Else], [FMP-ElseIf] und [/FMP-If] steuert das Tag [FMP-If] die HTML, die der Browser anzeigt.

CDML-Syntax: [FMP-If BooleanExpression]...[/FMP-If]

XSLT-CWP-Konvertierung:

- <xsl:choose><xsl:when test="BooleanExpression">...</xsl:when></xsl:choose>
- BooleanExpression wird zu XPath konvertiert.
- wenn in einem SGML-Tag/Attribut, <!—CDML Converter-FEHLER: [FMP-If] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	**[FMP-If: First Name .eq. field: Nick Name]** Your nick name is the same as your name. **[/FMP-If]**
Konvertierte XSLT-CWP:	**<xsl:choose><xsl:when test="'First Name' = fmrs:field[@name = 'Nick Name']/fmrs:data[1]">** Your nickname is the same as your name. **</xsl:when></xsl:choose>**
Konvertiertes Ergebnis:	Your nick name is the same as your name.

CDML-Tagname: Image (Bild)

Tag wird durch einen URL-Medienverweis auf das Bild im angegebenen Feld ersetzt.

CDML-Syntax: [FMP-Image: FieldName]

XSLT-CWP-Konvertierung:

- wenn aktueller Kontext ein Datensatz ist: <xsl:call-template name="get-image"><xsl:with-param name="fieldname" select="'fieldname'"/><xsl:with-param name="recid" select="$record/@record-id"/></xsl:call-template>
- sonst: <xsl:call-template name="get-image"><xsl:with-param name="fieldname" select="'fieldname'"/><xsl:with-param name="recid" select="$default-record/@record-id"/></xsl:call-template>

Konvertierungsbeispiel

Original-CDML:	<img src="**[FMP-Image: pictures]**">
Konvertierte XSLT-CWP:	<img> **<xsl:attribute name="src">** **<xsl:call-template name="get-image">** **<xsl:with-param name="fieldname" select="'pictures'"/>** **<xsl:with-param name="recid" select="$default-record/@record-id"/>** **</xsl:call-template>** **</xsl:attribute>** </img>
Konvertiertes Ergebnis:	**

CDML-Tagname: Einschließen

Tag wird durch den Inhalt einer anderen Datei ersetzt, gewöhnlich einer HTML-Formatdatei.

CDML-Syntax: [FMP-Include: FileName]

XSLT-CWP-Konvertierung:

- <!—CDML Converter-WARNUNG: Die Datei "FileName" wurde jetzt inline in dieses Dokument integriert. Der Bezug auf die ursprüngliche Datei wurde entfernt. Die ursprüngliche Datei wird nicht mehr benötigt. -->Konvertierter Inhalt der Datei
- wenn in einem SGML-Tag/Attribut, <!—CDML CONVERTER-FEHLER: [FMP-Include] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	[FMP-Include: requirefield.htm]
Konvertierte XSLT-CWP:	<!—CDML Converter-WARNUNG: Die Datei "requirefield.htm" wurde jetzt inline in dieses Dokument integriert. Der Bezug auf die ursprüngliche Datei wurde entfernt. Die ursprüngliche Datei wird nicht mehr benötigt. -->Konvertierter Inhalt der Datei
Konvertiertes Ergebnis:	Konvertierter Inhalt der Datei

CDML-Tagname: Include Field

Tag wird durch den Inhalt eines Felds ersetzt, gewöhnlich ein Textfeld, das eine HTML-Formatdatei enthält.

CDML-Syntax: [FMP-IncludeField: FieldName]

XSLT-CWP-Konvertierung:

- <!—CDML Converter-FEHLER: [FMP-IncludeField] wird nicht von XSLT-CWP unterstützt. -->

Konvertierungsbeispiel

Original-CDML:	[FMP-IncludeField: errorPage]
Konvertierte XSLT-CWP:	<!—CDML Converter-FEHLER: [FMP-IncludeField] wird nicht von XSLT-CWP unterstützt. -->
Konvertiertes Ergebnis:	<!—CDML Converter-FEHLER: [FMP-IncludeField] wird nicht von XSLT-CWP unterstützt. -->

CDML-Tagname: Inline Action (Inline-Aktion)

Tag erlaubt während der Verarbeitung einer einzelnen Formatdatei die Verarbeitung von mehreren CDML-Abfragen. Das Tag [FMP-InlineAction] nimmt das URL-ähnliche Format der Name-Wert-Paare für eine CDML-Abfrage als seine Parameter. Sämtliche weitere Verarbeitung der Formatdatei erfolgt dann, als hätte die Inline-Abfrage die Verarbeitung gestartet.

CDML-Syntax: [FMP-InlineAction: *intratags*]...[/FMP-InlineAction]

XSLT-CWP-Konvertierung:

- <xsl:variable name="inline-action" select="document(concat($authenticated-xml-base-uri, '/fmi/xml/fmresultset.xml?', '-db=', /fmrs:fmresultset/fmrs:datasource/@database, '&-lay=web3&title=s&-find'))"/><xsl:for-each select="$inline-action/fmrs:fmresultset/fmrs:resultset/fmrs:record"><xsl:variable name="inline-action-record" select="current()"/>...</xsl:for-each>
- wenn nicht weißer Leerinhalt zwischen dem Start- und End-Tag ist, Schleife über die Datensätze des Antwortdokuments
- die angeforderten URL-Parameter sind URL-kodiert
- der Abfrage-URL wird aus den *Intratags* erzeugt und ignoriert etwaige '–format'-Parameter
- wenn in einem SGML-Tag/Attribut, <!—CDML Converter-FEHLER: [FMP-InlineAction] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	**[FMP-InlineAction: -db={currentdatabase}, title="s", -lay=web3, find=]** Title:[FMP-Field: Title], Artist: [FMP-Field: Artist] [/FMP-InlineAction]
Konvertierte XSLT-CWP:	**<xsl:variable name="inline-action" select="document(concat($authenticated-xml-base-uri, '/fmi/xml/fmresultset.xml?',** **'-db=', /fmrs:fmresultset/fmrs:datasource/@database, '&-lay=web3&title=s&-find'))"/>** **<xsl:for-each select="$inline-action/fmrs:fmresultset/fmrs:resultset/fmrs:record">** **<xsl:variable name="inline-action-record" select="current()"/>** Title:<xsl:value-of select="**$inline-action-record/**fmrs:field[@name = 'Title']/fmrs:data[1]"/>, Artist: <xsl:value-of select="**$inline-action-record/**fmrs:field[@name = 'Artist']/fmrs:data[1]"/> </xsl:for-each>
Konvertiertes Ergebnis:	Titel: Mona Lisa, Artist: Leonardo da Vinci Titel: Sunflowers, Artist: Vincent Van Gogh

CDML-Tagname: Layout Fields (Layoutfelder)

Wiederholt die HTML zwischen [FMP-LayoutFields] und [/FMP-LayoutFields] für alle Felder im angegebenen Layout, das Teil der Abfrage war, die diese Seite erzeugt hat.

CDML-Syntax: [FMP-LayoutFields]...[/FMP-LayoutFields]

XSLT-CWP-Konvertierung:

- <xsl:for-each select="fmrs:fmresultset/fmrs:metadata/fmrs:field-definition" />...</xsl:for-each>
- wenn in einem SGML-Tag/Attribut, <!—CDML Converter-FEHLER: [FMP-LayoutFields] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	<select name="-SortField"> <option value="">-None- **[FMP-LayoutFields]** <option>[FMP-FieldName: Raw] **[/FMP-LayoutFields]** </select>
Konvertierte XSLT-CWP:	<select name="-SortField.1"> <option value="" />-None-</option> **<xsl:for-each select="fmrs:fmresultset/fmrs:metadata/fmrs:field-definition" />** <option /><xsl:value-of select="@name" /> **</xsl:for-each>** </select>
Konvertiertes Ergebnis:	<select name="-SortField.1"> <option value="">-None- <option >First Name</option> <option >Last Name</option> <option >Employee Number</option> </select>

CDML-Tagname: Link

Tag wird durch einen URL ersetzt, der auf die Seite verweist, auf der er sich befindet. Die Parameter können verwendet werden, um Teile des generierten URLs zu entfernen.

CDML-Syntax: <sgml_tag sgml_attr="[FMP-Link: *CharacterCodes*]&-format=*FileName*.htm &name1=*value1*&name2=*value2*...">

XSLT-CWP-Konvertierung:

- Einfügung: <xsl:call-template name="get-link"><xsl:with-param name="filter-codes" select="'*CharacterCodes*'"/> <xsl:with-param name="*stylesheet*" select="'*FileName*.xsl'"/><xsl:with-param name="*other-params*" select="'&name1=*value1*&name2=*value2*...'"/></xsl:call-template>
- wenn in einem SGML-Attribut und gefolgt von einem '–format'-Parameter: Der *'stylesheet'*-Parameter wird an die 'get-link'-Vorlage übergeben.
- wenn in einem SGML-Attribut und gefolgt von einem zusätzlichen Query-Parameter: Der *'other-params'*-Parameter wird an die 'get-link'-Vorlage übergeben.

Konvertierungsbeispiel

Original-CDML:	Alternate hit list
Konvertierte XSLT-CWP:	<a> **<xsl:attribute name="href">** **<xsl:call-template name="get-link">** **<xsl:with-param name="filter-codes" select="'adr'"/>** **<xsl:with-param name="stylesheet" select="'AltHitList.xsl'"/>** **<xsl:with-param name="other-params" select="'&-db=art'"/>** **</xsl:call-template>** **</xsl:attribute>**Alternate hit list
Konvertiertes Ergebnis:	Alternate hit list

CDML-Tagname: Link First (Link auf ersten Bereich)

Die HTML zwischen [FMP-LinkFirst] und [/FMP-LinkFirst] wird durch einen Link auf den ersten Datensatzbereich ersetzt, basierend auf dem -Max-Wert, der zur Erstellung der aktuellen Seite verwendet wurde.

CDML-Syntax: [FMP-LinkFirst]...[/FMP-LinkFirst]

XSLT-CWP-Konvertierung:

- ...
- Einfügung in oberster Dokumentebene: <xsl:variable name="link-first"><xsl:call-template name="get-link-first"/></xsl:variable>
- wenn in einem SGML-Tag/Attribut: <!—CDML Converter-FEHLER: [FMP-LinkFirst] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	**[FMP-LinkFirst]**First set of records**[/FMP-LinkFirst]**
Konvertierte XSLT-CWP:	****First set of records****
Konvertiertes Ergebnis:	****First set of records****

CDML-Tagname: Link Last (Link auf letzten Bereich)

Die HTML zwischen [FMP-LinkLast] und [/FMP-LinkLast] wird durch einen Link auf den letzten Datensatzbereich ersetzt, basierend auf dem -Max-Wert, der zur Erstellung der aktuellen Seite verwendet wurde.

CDML-Syntax: [FMP-LinkLast]...[/FMP-LinkLast]

XSLT-CWP-Konvertierung:

- ...
- Einfügung in oberster Dokumentebene: <xsl:variable name="link-last"><xsl:call-template name="get-link-last"/>
  </xsl:variable>
- wenn in einem SGML-Tag/Attribut: <!—CDML Converter-FEHLER: [FMP-LinkLast] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	**[FMP-LinkLast]**Last set of records**[/FMP-LinkLast]**
Konvertierte XSLT-CWP:	****Last set of records****
Konvertiertes Ergebnis:	****Last set of records****

CDML-Tagname: Link Next (Link auf nächsten Bereich)

Die HTML zwischen [FMP-LinkNext] und [/FMP-LinkNext] wird durch einen Link auf den nächsten Datensatzbereich ersetzt, basierend auf dem –Max- und dem –Skip-Wert, die zur Erstellung der aktuellen Seite verwendet wurden.

CDML-Syntax: [FMP-LinkNext]...[/FMP-LinkNext]

XSLT-CWP-Konvertierung:

- ...
- Einfügung in oberster Dokumentebene: <xsl:variable name="link-next"><xsl:call-template name="get-link-next"/>
  </xsl:variable>
- wenn in einem SGML-Tag/Attribut: <!—CDML Converter-FEHLER: [FMP-LinkNext] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	**[**FMP-LinkNext]Next set of records**[/FMP-LinkNext]**
Konvertierte XSLT-CWP:	****Next set of records****
Konvertiertes Ergebnis:	****Next set of records****

CDML-Tagname: Link Previous (Link auf vorherigen Bereich)

Die HTML zwischen [FMP-LinkPrevious] und [/FMP-LinkPrevious] wird durch einen Link auf den vorherigen Datensatzbereich ersetzt, basierend auf dem –Max- und dem –Skip-Wert, die zur Erstellung der aktuellen Seite verwendet wurden.

CDML-Syntax: [FMP-LinkPrevious]...[/FMP-LinkPrevious]

XSLT-CWP-Konvertierung:

- ...
- Einfügung in oberster Dokumentebene: <xsl:variable name="link-previous">
  <xsl:call-template name="get-link-previous"/></xsl:variable>
- wenn in einem SGML-Tag/Attribut: <!—CDML Converter-FEHLER: [FMP-LinkPrevious] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	**[FMP-LinkPrevious]**Previous set of records**[/FMP-LinkPrevious]**
Konvertierte XSLT-CWP:	****Previous set of records****
Konvertiertes Ergebnis:	****Previous set of records****

CDML-Tagname: Link to Record ID (Link auf Datensatz-ID)

Tag wird durch einen URL auf einen bestimmten Datensatz in einer Datenbank ersetzt. Der generierte Link umfasst alle Such- und Sortierkriterien, die zur Erstellung der Seite, auf der sich dieses Tag befindet, benötigt wurde. Der Layout-Parameter ist optional.

CDML-Syntax: [FMP-LinkRecID: Format=*FileName*, Layout=*LayoutName*]

XSLT-CWP-Konvertierung:

- <xsl:call-template name="get-link-rec-id"><xsl:with-param name="rec-id" select="@record-id"/><xsl:with-param name="stylesheet" select="'/FileName.xsl'"/></xsl:call-template>
- wenn von einem '–format'-Parameter gefolgt: Der 'stylesheet'-Parameter wird an die 'get-link-rec-id'-Vorlage übergeben.
- wenn von einem '–lay'-Parameter gefolgt: Der 'layout'-Parameter wird an die 'get-link-rec-id'-Vorlage übergeben.
- wenn nicht in einem SGML-Attribut: <!—CDML Converter-FEHLER: [FMP-LinkRecID] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	More detail
Konvertierte XSLT-CWP:	<a> <xsl:attribute name="href"> **<xsl:call-template name="get-link-rec-id">** **<xsl:with-param name="rec-id" select="$record/@record-id"/>** **<xsl:with-param name="stylesheet" select="'FormatFile.xsl'"/>** **<xsl:with-param name="layout" select="'detail'"/>** **</xsl:call-template>** </xsl:attribute>More Detail
Konvertiertes Ergebnis:	More detail

CDML-Tagname: Log (Protokoll)

Tag wird durch den Text ersetzt, der als der Parameter angegeben und als Meldungen des Benutzerprotokolls in die Programmprotokolldatei geschrieben wurde. Weitere Informationen finden Sie unter „Verwenden des Web Publishing Engine-Programmprotokolls“ auf Seite 93.

CDML-Syntax: [FMP-Log: *einText*]

XSLT-CWP-Konvertierung:

- <xsl:message>*someText*</xsl:message>
- wenn in einem SGML-Tag/Attribut: <!—CDML Converter-FEHLER: [FMP-Log] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	Log code is here: **[FMP-Log: the search page was accessed]**
Konvertierte XSLT-CWP:	Log code is here: **<xsl:message>the search page was accessed</xsl:message>**
Konvertiertes Ergebnis:	Log code is here: (Der Text **the search page was accessed** wird in das Programmprotokoll geschrieben, wenn die Benutzerprotokollierung aktiviert ist.)

CDML-Tagname: Option

Tag wird durch alle Werte in der Werteliste des Felds in diesem Layout ersetzt.

CDML-Syntax: [FMP-Option: FieldName, List=*ValueListName*]

Hinweis Für eine korrekte Konvertierung benötigt das CDML-Tag "[FMP-Option]" Felder im Layout, die auf die angegebene(n) Werteliste(n) verweisen.

XSLT-CWP-Konvertierung:

- <xsl:variable name="valuelist-name" select="$layout/fml:FMPXMLLAYOUT/fml:LAYOUT/fml:FIELD[@NAME = 'field3']/fml:STYLE/@VALUELIST"/><xsl:for-each select="$layout/fml:FMPXMLLAYOUT/fml:VALUELISTS/fml:VALUELIST[@NAME = $valuelist-name-4]/fml:VALUE"><option value="{current()}"><xsl:if test="current() = $default-record/fmrs:field[@name = 'field3']/fmrs:data[1]"><xsl:attribute name="selected">selected</xsl:attribute></xsl:if><xsl:value-of select="current()"/></option>
  </xsl:for-each>
  - wenn aktueller Kontext ein Datensatz ist: Verwenden Sie fmrs:field[@name = 'FieldName']/fmrs:data[1] oben.
  - sonst: Verwenden Sie /fmrs:fmresultset/fmrs:resultset/fmrs:record[1]/fmrs:field[@name = 'FieldName']/fmrs:data[1] oben.

- Parameter '*List*' wird ignoriert: <!--CDML2XSLT-WARNUNG: [FMP-Option] Der Parameter 'List' wird nicht von XSLT-CWP unterstützt. -->
- wenn in einem SGML-Tag/Attribut: <!—CDML Converter-FEHLER: [FMP-Option] war nicht richtig verschachtelt -->
- wenn im Dokument benutzt: $layout-Variable wird in oberster Dokumentebene erstellt.

Konvertierungsbeispiel

Original-CDML:	<select name="Groups"> **[FMP-option: Groups]** </select>
Konvertierte XSLT-CWP:	<select> **<xsl:variable name="valuelist-name" select="$layout/fml:FMPXMLLAYOUT/ fml:LAYOUT/fml:FIELD[@NAME = 'Groups']/fml:STYLE/@VALUELIST"/>** **<xsl:for-each select="$layout/fml:FMPXMLLAYOUT/fml:VALUELISTS/ fml:VALUELIST[@NAME = $valuelist-name]/fml:VALUE">** **<option value="{current()}"><xsl:if test="current() = $default-record/fmrs:field[@name = 'Groups']/fmrs:data[1]"><xsl:attribute name="selected">selected</xsl:attribute></ xsl:if><xsl:value-of select="current()"/></option></xsl:for-each>** </select>
Konvertiertes Ergebnis:	<select name="Groups"> **<option> Production** **<option selected> Sales** **<option> Support** </select>

CDML-Tagname: Ausschnitt

Wiederholt die HTML zwischen [FMP-Portal] und [/FMP-Portal] für jeden Datensatz in einem Ausschnitt.

CDML-Syntax: [FMP-Portal: RelationshipName]...[/FMP-Portal]

XSLT-CWP-Konvertierung:

- wenn aktueller Kontext ein Datensatz ist: <xsl:for-each select="fmrs:relatedset[@table = 'RelationshipName']/fmrs:record">...</xsl:for-each>
- sonst: <xsl:for-each select="/fmrs:fmresultset/fmrs:resultset/fmrs:record/fmrs:relatedset[@table='RelationshipName']/fmrs:record">...</xsl:for-each>
- wenn in einem SGML-Tag/Attribut: <!—CDML Converter-FEHLER: [FMP-Portal] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	**[FMP-Portal: lineitems]** [FMP-CurrentPortalRowNumber]: [FMP-Field: lineitems::name] [/FMP-Portal]
Konvertierte XSLT-CWP:	**<xsl:for-each select="fmrs:relatedset[@table='lineitems']/fmrs:record">** <xsl:value-of select="position()" />: <xsl:value-of select="fmrs:field[@name = 'name']" /> </xsl:for-each>
Konvertiertes Ergebnis:	1: Red

CDML-Tagname: Range End (Bereichsende)

Tag wird durch die Nummer des zuletzt angezeigten Datensatzes ersetzt.

CDML-Syntax: [FMP-RangeEnd]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {$current-skip + /fmrs:fmresultset/fmrs:resultset/@fetch-size}
- sonst: <xsl:value-of select="$current-skip + /fmrs:fmresultset/fmrs:resultset/@fetch-size"/>
- wenn im Dokument benutzt: $current-skip-Variable wird mithilfe der benannten Vorlage des Stylesheets cdml2xsl_utilities.xsl auf der obersten Ebene erstellt.

Konvertierungsbeispiel

Original-CDML:	Records [FMP-RangeStart] through **[FMP-RangeEnd]**
Konvertierte XSLT-CWP:	Records <xsl:value-of select="$current-skip + 1"/> through **<xsl:value-of select="$current-skip + /fmrs:fmresultset/fmrs:resultset/@fetch-size"/>**
Konvertiertes Ergebnis:	Records 6 through **10**

CDML-Tagname: Range Size (Bereichsgröße)

Tag wird durch die Anzahl der Datensätze ersetzt, die auf der Seite angezeigt wird.

CDML-Syntax: [FMP-RangeSize]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {/fmrs:fmresultset/fmrs:resultset/@fetch-size}
- sonst: <xsl:value-of select="/fmrs:fmresultset/fmrs:resultset/@fetch-size" />

Konvertierungsbeispiel

Original-CDML:	You are viewing **[FMP-RangeSize]** records.
Konvertierte XSLT-CWP:	You are viewing **<xsl:value-of select="/fmrs:fmresultset/fmrs:resultset/@fetch-size" />** records.
Konvertiertes Ergebnis:	You are viewing **8** records.

CDML-Tagname: Range Start (Bereichsanfang)

Tag wird durch die Nummer des ersten angezeigten Datensatzes ersetzt.

CDML-Syntax: [FMP-RangeStart]

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {$current-skip + 1}
- sonst: <xsl:value-of select="$current-skip + 1"/>
- wenn im Dokument benutzt: $current-skip-Variable wird mithilfe der benannten Vorlage des Stylesheets cdml2xsl_utilities.xsl auf der obersten Ebene erstellt.

Konvertierungsbeispiel

Original-CDML:	Records **[FMP-RangeStart]** through [FMP-RangeEnd]
Konvertierte XSLT-CWP:	Records **<xsl:value-of select="$current-skip + 1"/>** through <xsl:value-of select="$current-skip + /fmrs:fmresultset/fmrs:resultset/@fetch-size"/>
Konvertiertes Ergebnis:	Records **6** through 10

CDML-Tagname: Datensatz

Wiederholt die HTML zwischen [FMP-Record] und [/FMP-Record] für jeden Datensatz bis zu –Max, beginnend nach dem Überspringen von –Skip-Datensätzen.

CDML-Syntax: [FMP-Record]...[/FMP-Record]

XSLT-CWP-Konvertierung:

- <xsl:for-each select="/fmrs:fmresultset/fmrs:resultset/fmrs:record">...</xsl:for-each>
- wenn in einem SGML-Tag/Attribut: <!—CDML Converter-FEHLER: [FMP-Record] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	**[FMP-Record]** [FMP-Field: Country] - [FMP-Field: Capital] **[/FMP-Record]**
Konvertierte XSLT-CWP:	**<xsl:for-each select="/fmrs:fmresultset/fmrs:resultset/fmrs:record">** <xsl:value-of select="fmrs:field[@name = 'Country']/fmrs:data[1]" /> - <xsl:value-of select="fmrs:field[@name = 'Capital']/fmrs:data[1]" /> **</xsl:for-each>**
Konvertiertes Ergebnis:	Great Britain - London France - Paris USA - Washington D.C.

CDML-Tagname: Repeating (Wiederholen)

Wiederholt die HTML zwischen [FMP-Repeating] und [/FMP-Repeating] für alle Wiederholungen des angegebenen Felds. Tag [FMP-RepeatingItem] wird durch den Inhalt einer bestimmten Wiederholung ersetzt.

CDML-Syntax: [FMP-Repeating: *FieldName*]...[/FMP-Repeating]

XSLT-CWP-Konvertierung:

- wenn aktueller Kontext ein Datensatz ist: <xsl:for-each select="fmrs:field[@name = '*FieldName*']/fmrs:data">...</xsl:for-each>
- sonst: <xsl:for-each select="/fmrs:fmresultset/fmrs:resultset/fmrs:record[1]/fmrs:field[@name = '*FieldName*']/fmrs:data">...</xsl:for-each>
- wenn in einem SGML-Tag/Attribut: <!—CDML Converter-FEHLER: [FMP-Repeating] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	**[FMP-Repeating: extensions]** [FMP-CurrentRepeatNumber]: [FMP-RepeatingItem] **[/FMP-Repeating]**
Konvertierte XSLT-CWP:	**<xsl:for-each select="fmrs:field[@name = 'extensions']/fmrs:data">** <xsl:value-of select="position()" />: <xsl:value-of select="." /> **</xsl:for-each>**
Konvertiertes Ergebnis:	3: Green

CDML-Tagname: Repeating Item (Wiederholobjekt)

Tag wird durch den Inhalt der nächsten Wiederholung ersetzt. Diese befindet sich immer in einer [FMP-Repeating]-Schleife.

CDML-Syntax: [FMP-RepeatingItem: *Kodierung*], wobei *Kodierung* 'Raw', 'URL', 'HTML' (Standard) oder 'Break' ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {.}
- sonst: <xsl:value-of select="." />

Konvertierungsbeispiel

Original-CDML:	[FMP-Repeating: extensions] [FMP-CurrentRepeatNumber]: **[FMP-RepeatingItem]** [/FMP-Repeating]
Konvertierte XSLT-CWP:	<xsl:for-each select="fmrs:field[@name = 'extensions']/fmrs:data"> <xsl:value-of select="position()" />: **<xsl:value-of select="." />** </xsl:for-each>
Konvertiertes Ergebnis:	3: **Green**

CDML-Tagname: Set Cookie (Cookie setzen)

Tag wird in der HTML mit nichts ersetzt und wird verwendet, um eine Variable im Browser des Benutzers zu speichern. Die Parameter Expires, Path und Domain sind optional.

CDML-Syntax: [FMP-SetCookie: name=*value*, Expires=*expires*, Path=*path*, Domain=*domain*]

XSLT-CWP-Konvertierung:

- <xsl:variable name="cookie-*name*" select="fmxslt:set_cookie('*name*', '*value*', '*expires*', '*path*', '*domain*')" />
- undefinierte optionale Parameter *expires*, *path*, *domain* werden als leere Strings übergeben'.
- wenn in einem SGML-Tag/Attribut: <!—CDML Converter-FEHLER: [FMP-SetCookie] war nicht richtig verschachtelt -->

Konvertierungsbeispiel

Original-CDML:	Set-Cookie here: **[FMP-SetCookie: ColorChoice=Green, Expires=43200]**
Konvertierte XSLT-CWP:	Set-Cookie here: **<xsl:variable name="cookie-ColorChoice" select="fmxslt:set_cookie('ColorChoice', 'Value', '43200', '', '')" />**
Konvertiertes Ergebnis:	Set-Cookie here:

CDML-Tagname: Sort Field Item (Sortierfeldobjekt)

Tag wird durch den Feldnamen ersetzt, der Teil der Abfrage für die Erstellung dieser Seite war. Dieser befindet sich immer in einer [FMP-CurrentSort]-Schleife.

CDML-Syntax: [FMP-SortFieldItem: *Kodierung*], wobei *kodierung* 'Raw', 'URL' oder 'HTML' (Standard) ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {@name}
- sonst: <xsl:value-of select="@name" />
- wenn im Dokument benutzt: $current-sort-Variable wird mithilfe der benannten Vorlage des Stylesheets cdml2xsl_utilities.xsl auf der obersten Ebene erstellt.
- wenn außerhalb einer [FMP-CurrentSort]-Schleife: <!—CDML Converter-FEHLER: [FMP-FindFieldItem] außerhalb von [FMP-CurrentFind] -->

Konvertierungsbeispiel

Original-CDML:	Aktuelle Sortierfolge: [FMP-CurrentSort] Field: **[FMP-SortFieldItem]**, Order: [FMP-SortOrderItem] [/FMP-CurrentSort]
Konvertierte XSLT-CWP:	Aktuelle Sortierfolge: <xsl:for-each select="xalan:nodeset($current-find)/find-field"> Field: **<xsl:value-of select="@name" />**, Order: <xsl:value-of select="@order" /> </xsl:for-each>
Konvertiertes Ergebnis:	Aktuelle Sortierfolge: Field: **First Name**, Order: descend Field: **First Name**, Order: descend

CDML-Tagname: Sort Order Item (Sortierfolgenobjekt)

Tag wird durch die Sortierfolge ersetzt, die Teil der Abfrage für die Erstellung dieser Seite war. Dieser befindet sich immer in einer [FMP-CurrentSort]-Schleife.

CDML-Syntax: [FMP-SortOrderItem: *Kodierung*], wobei *Kodierung* einen der folgenden Werte hat:

- 'Raw'
- 'URL'
- 'HTML' (Standard)
- 'Display'

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {@order}
- sonst: <xsl:value-of select="@order" />
- wenn *Kodierung* gleich 'Display' ist, wird 'HTML' verwendet: <!—CDML2XSLT-WARNUNG: [FMP-SortOrderItem] Die Kodierung 'Display' wird von XSLT-CWP nicht unterstützt. -->
- wenn im Dokument benutzt: $current-sort-Variable wird mithilfe der benannten Vorlage des Stylesheets cdml2xsl_utilities.xsl auf der obersten Ebene erstellt.
- wenn außerhalb einer [FMP-CurrentSort]-Schleife: <!—CDML Converter-FEHLER: [FMP-SortOrderItem] außerhalb von [FMP-CurrentSort] -->

Konvertierungsbeispiel

Original-CDML:	Aktuelle Sortierfolge: [FMP-CurrentSort] Field: [FMP-SortFieldItem], Order: **[FMP-SortOrderItem]** [/FMP-CurrentSort]
Konvertierte XSLT-CWP:	Aktuelle Sortierfolge: <xsl:for-each select="xalan:nodeset($current-find)/find-field"> Field: <xsl:value-of select="@name" />, Order: **<xsl:value-of select="@order" />** </xsl:for-each>
Konvertiertes Ergebnis:	Aktuelle Sortierfolge: Field: First Name, Order: **descend** Field: Last Name, Order: **descend**

CDML-Tagname: Value List (Werteliste)

Wiederholt die HTML zwischen [FMP-ValueList] und [/FMP-ValueList] für alle Werte in der angegebenen Werteliste.

CDML-Syntax: [FMP-ValueList: FieldName, List=*ValueListName*]...[/FMP-ValueList]

Hinweis Für eine korrekte Konvertierung benötigt das CDML-Tag "[FMP-ValueList]" Felder im Layout, die auf die angegebene(n) Werteliste(n) verweisen.

XSLT-CWP-Konvertierung:

- <xsl:variable name="valuelist-name" select="$layout/fml:FMPXMLLAYOUT/fml:LAYOUT/fml:FIELD[@NAME = 'FieldName']/fml:STYLE/@VALUELIST"/><xsl:for-each select="$layout/fml:FMPXMLLAYOUT/fml:VALUELISTS/fml:VALUELIST[@NAME = $valuelist-name]/fml:VALUE">...</xsl:for-each>
- Parameter 'List' wird ignoriert: <!—CDML2XSLT-WARNUNG: [FMP-ValueList] Der Parameter 'List' wird nicht von XSLT-CWP unterstützt. -->
- wenn in einem SGML-Tag/Attribut: <!—CDML Converter-FEHLER: [FMP-ValueList] war nicht richtig verschachtelt -->
- wenn im Dokument benutzt: $layout-Variable wird in oberster Dokumentebene erstellt.

Konvertierungsbeispiel

Original-CDML:	**[FMP-ValueList: Groups, List=GroupList]** <input type="radio" name="Groups" value="[FMP-ValueListItem]" [FMP-ValueListChecked]>[FMP-ValueListItem] [/FMP-ValueList]
Konvertierte XSLT-CWP:	**<xsl:variable name="valuelist-name" select="$layout/fml:FMPXMLLAYOUT/fml:LAYOUT/fml:FIELD[@NAME = 'Groups']/fml:STYLE/@VALUELIST"/>** **<xsl:for-each select="$layout/fml:FMPXMLLAYOUT/fml:VALUELISTS/fml:VALUELIST[@NAME = $valuelist-name]/fml:VALUE">** **<!—CDML2XSLT-WARNUNG: [FMP-ValueList] Der Parameter 'List' wird nicht von XSLT-CWP unterstützt. -->** <input type="radio" name="Groups" value="{current()}"><xsl:if test=". = $current-record/fmrs:field[@name = 'Groups']/fmrs:data[1]"><xsl:attribute name="checked">checked</xsl:attribute></xsl:if></input><xsl:value-of select="{current()}" />
Konvertiertes Ergebnis:	<input type="radio" name="Groups" value="Production">Production <input type="radio" name="Groups" value="Sales" checked>Sales <input type="radio" name="Groups" value="Support">Support

CDML-Tagname: Value List Checked (Werteliste-Markierung)

Tag wird für jedes Wertelistenobjekt, das im angegebenen Feld ausgewählt wurde, durch das Wort "checked" ersetzt. Dieser befindet sich immer in einer [FMP-ValueList]-Schleife.

CDML-Syntax: <input name="fieldname" value="fieldvalue" [FMP-ValueListChecked]>

XSLT-CWP-Konvertierung:

- <xsl:if test="current() = fmrs:field[@name = 'fieldname']/fmrs:data[1]"><xsl:attribute name="checked">checked </xsl:attribute></xsl:if>
- wenn außerhalb einer [FMP-ValueList]-Schleife: <!—CDML Converter-FEHLER: [FMP-ValueListChecked] außerhalb von [FMP-ValueList] -->
- wenn nicht in einem SGML-'input'-Tag: <!—CDML Converter-FEHLER: [FMP-ValueListChecked] war nicht richtig verschachtelt -->
- wenn im Dokument benutzt: $layout-Variable wird in oberster Dokumentebene erstellt.

Konvertierungsbeispiel

Original-CDML:	[FMP-ValueList: Groups, List=GroupList] <input type="radio" name="Groups" value="[FMP-ValueListItem]" **[FMP-ValueListChecked]**>[FMP-ValueListItem] [/FMP-ValueList]

Konvertierungsbeispiel

Konvertierte XSLT-CWP:	<xsl:variable name="valuelist-name" select="$layout/fml:FMPXMLLAYOUT/fml:LAYOUT/fml:FIELD[@NAME = 'Groups']/fml:STYLE/@VALUELIST"/> <xsl:for-each select="$layout/fml:FMPXMLLAYOUT/fml:VALUELISTS/fml:VALUELIST[@NAME = $valuelist-name]/fml:VALUE"> <!—CDML2XSLT-WARNUNG: [FMP-ValueList] Der Parameter 'List' wird nicht von XSLT-CWP unterstützt. --> <input type="radio" name="Groups" value="{current()}"><**xsl:if test="current() = fmrs:field[@name = 'Groups']/fmrs:data[1]">** **<xsl:attribute name="checked">checked</xsl:attribute></xsl:if>**</input><xsl:value-of select="{current()}" /> </xsl:for-each>
Konvertiertes Ergebnis:	<input type="radio" name="Groups" value="Production">Production <input type="radio" name="Groups" value="Sales" **checked**>Sales <input type="radio" name="Groups" value="Support">Support

CDML-Tagname: Value List Item (Werteliste neintrag)

Tag wird durch den nächsten Eintrag in einer Werteliste ersetzt. Dieser befindet sich immer in einer [FMP-ValueList]-Schleife.

CDML-Syntax: [FMP-ValueListItem: *Checked, Kodierung*], wobei *Kodierung* 'Raw', 'URL', or 'HTML' (Standard) ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {current()}
- sonst: <xsl:value-of select="current()" />
- wenn außerhalb einer [FMP-ValueList]-Schleife: <!—CDML Converter-FEHLER: [FMP-ValueListItem] außerhalb von [FMP-ValueList] -->
- wenn im Dokument benutzt: $layout-Variable wird in oberster Dokumentebene erstellt.

Konvertierungsbeispiel

Original-CDML:	[FMP-ValueList: Groups, List=GroupList] <input type="radio" name="Groups" value="**[FMP-ValueListItem]**" [FMP-ValueListChecked]> **[FMP-ValueListItem]** [/FMP-ValueList]

Konvertierungsbeispiel

Konvertierte XSLT-CWP:	<xsl:variable name="valuelist-name" select="$layout/fml:FMPXMLLAYOUT/fml:LAYOUT/fml:FIELD[@NAME = 'Groups']/fml:STYLE/@VALUELIST"/> <xsl:for-each select="$layout/fml:FMPXMLLAYOUT/fml:VALUELISTS/fml:VALUELIST[@NAME = $valuelist-name]/fml:VALUE"> <!—CDML2XSLT-WARNUNG: [FMP-ValueList] Der Parameter 'List' wird nicht von XSLT-CWP unterstützt. --> <input type="radio" name="Groups" value="**{current()}**"><xsl:if test=". = $current-record/fmrs:field[@name = 'Groups']/fmrs:data[1]"><xsl:attribute name="checked">checked</xsl:attribute></xsl:if></input>**<xsl:value-of select="{current()}" />**
Konvertiertes Ergebnis:	<input type="radio" name="Groups" value="**Production**">**Production** <input type="radio" name="Groups" value="**Sales**" checked>**Sales** <input type="radio" name="Groups" value="**Support**">**Support**

CDML-Tagname: Value Name Item (Wertelistenname)

Tag wird durch den Namen einer Werteliste ersetzt. Dieser befindet sich immer in einer [FMP-ValueNames]-Schleife.

CDML-Syntax: [FMP-ValueNameItem: *Kodierung*], wobei *kodierung* 'Raw', 'URL' oder 'HTML' (Standard) ist.

XSLT-CWP-Konvertierung:

- wenn innerhalb des SGML-Element-Attributwerts: {@NAME}
- sonst: <xsl:value-of select="@NAME" />
- wenn in einem SGML-Tag/Attribut: <!—CDML Converter-FEHLER: [FMP-ValueNames] war nicht richtig verschachtelt -->
- wenn außerhalb einer [FMP-ValueNames]-Schleife: <!—CDML Converter-FEHLER: [FMP-ValueNameItem] außerhalb von [FMP-ValueNames] -->
- wenn im Dokument benutzt: $layout-Variable wird in oberster Dokumentebene erstellt.

Konvertierungsbeispiel

Original-CDML:	<select name="–sortorder"> <option>Ascending <option>Descending [FMP-ValueNames] <option value="Custom=[FMP-ValueNameItem]">**[FMP-ValueNameItem]** [/FMP-ValueNames] </select>

Konvertierungsbeispiel

Konvertierte XSLT-CWP:	<select name="–sortorder.1"> <option />Ascending <option />Descending <xsl:for-each select="$layout/fml:fmpxmllayout/fml:VALUELISTS/fml:VALUELIST"> <option value="Custom=**{@NAME}**" />**<xsl:value-of select="@NAME" />** /xsl:for-each> </select>
Konvertiertes Ergebnis:	<select name="–sortorder.1"> <option />Ascending <option />Descending <option value="Custom=Colors">**Colors** <option value="Custom=Sizes">**Sizes** </select>

CDML-Tagname: Value Names (Wertenamen)

Wiederholt die HTML zwischen [FMP-ValueNames] und [/FMP-ValueNames] für alle Wertelisten in der Datenbank.

CDML-Syntax: [FMP-ValueNames]...[/FMP-ValueNames]

Hinweis Für eine korrekte Konvertierung benötigt das CDML-Tag "[FMP-ValueNames]" Felder im Layout, die auf die angegebene(n) Werteliste(n) verweisen.

XSLT-CWP-Konvertierung:

- <xsl:for-each select="$layout/fml:fmpxmllayout/fml:VALUELISTS/fml:VALUELIST">...</xsl:for-each>
- wenn in einem SGML-Tag/Attribut: <!—CDML Converter-FEHLER: [FMP-ValueNames] war nicht richtig verschachtelt -->
- wenn im Dokument benutzt: $layout-Variable wird in oberster Dokumentebene erstellt.

Konvertierungsbeispiel

Original-CDML:	<select name="–sortorder"> <option>Ascending <option>Descending **[FMP-ValueNames]** <option value="Custom=[FMP-ValueNameItem]">[FMP-ValueNameItem] **[/FMP-ValueNames]** </select>

Konvertierungsbeispiel

Konvertierte XSLT-CWP:	<select name="–sortorder.1"> <option />Ascending <option />Descending **<xsl:for-each select="$layout/fml:fmpxmllayout/fml:VALUELISTS/fml:VALUELIST">** <option value="Custom={@NAME}" /><xsl:value-of select="@NAME" /> **</xsl:for-each>** </select>
Konvertiertes Ergebnis:	<select name="–sortorder.1"> <option />Ascending <option />Descending <option value="Custom=Colors">Colors <option value="Custom=Sizes">Sizes </select>

Index

A

Abfragen von XML-Daten 27
Abrufen verfügbarer Scriptnamen 102
Administrationskonsole für Web Publishing Engine 20–27, 47
Aktivieren von Instant Web Publishing in Datenbank 21
Angeben der XML-Grammatik 60
ASCII-Zeichen, in XML-Dokumenten 40
Ausblenden von Metadaten mit Stylesheets 45
Ausgabe von Daten mit Stylesheets 45
Ausgabeseiten
  Anfängliche Standardkodierungseinstellung 64
  Ausgabemethode, angeben 64
  Kodierung, angeben 64
  <xsl:output>-Element 64
Ausschnitte
  Datensätze bearbeiten 99
  Datensätze hinzufügen 98
authenticated base URI parameter 68
Authentifizierung von Web-Benutzern 21, 69
Authentifizierung Web-Benutzer 21, 67
Authentifizierung Web-Benutzern 67

B

Basis-URI-Parameter 68
Befehle, Query s. Query-Strings
Beispiele für
  Generierte FMPXMLLAYOUT-Grammatik 39
  Generierte FMPXMLRESULT-Grammatik 37
  Generierte fmresultset-Grammatik 35
Benutzernamen
  Authentifizierung Web-Benutzer 21, 67
  Für Custom Web Publishing definieren 21
  Zugriff auf XML-Dokumente 69
Berechtigung, Zuweisen für Custom Web Publishing 21
break_encode(), Erweiterungsfunktion 78

C

CDML Converter
  Beschreibung 14, 50, 125
  Generierte Stylesheets korrigieren 53, 125
  Generierte Stylesheets, testen 54
  Generierte Stylesheets, verwenden 54
  Installieren 51
  Starten und verwenden 51
CDML-Konvertierung zu XSLT
  Beschreibung 125
  Konvertierungsfehler beheben 53, 135
  Prozess 125
  Unterstützte Aktions-Tags konvertieren 130
  Unterstützte Variablen-Tags konvertieren 131
  Veraltete Aktions-Tags konvertieren 130
  Veraltete Variablen-Tags konvertieren 131
  Zuordnungsregeln 53, 125
check_error_status(), Erweiterungsfunktion 85, 121
Clientinformationen, über XSLT-Parameter abrufen 67
Client-seitige Stylesheets 27, 43
compare_date(), Erweiterungsfunktion 82
compare_datetime(), Erweiterungsfunktion 83
compare_day(), Erweiterungsfunktion 82
compare_time(), Erweiterungsfunktion 82
contains_checkbox_value(), Erweiterungsfunktion 80
convert_datetime(), Erweiterungsfunktion 83
Cookies
  Erweiterungsfunktionen, verwenden 76
  Sitzungs-ID speichern 71
create_session(), Erweiterungsfunktion 72

Custom Web Publishing
Beschreibung 11, 14
Erweitertes Zugriffsrecht für 21
Gastkonto 22
In Datenbank aktivieren 21
In Web Publishing Engine aktivieren 22
Mit XSLT 46, 57
Neue Funktionen in 16
Scripts 18
Scripts verwenden 17
Statische IP-Adresse verwenden 15
Voraussetzungen für das 15
XML verwenden 25
Zugriff auf Lösungen von Web-Benutzern 21, 69
Zugriff von IP-Adressen in Web-Server beschränken 23
Zugriff Web-Benutzer-Lösungen 67
Custom Web Publishing Engine (CWPE) 26, 46

D

Element <datasource> 33
Datenbanken, veröffentlichte schützen 22
Datenbankfehlercodes 32
Datenbanksitzungen, aktivieren 73, 100
Datums-Erweiterungsfunktionen, verwenden 81
Datumsformatstrings 83
–db (Query-Parameter) 103
–dbnames (Query-Befehl) 100
Definieren von Erweiterungsfunktionen 86
–delete (Query-Befehl) 100
Document Type Definitions (DTDs) 33, 36
document(), Funktion 68
Dokumentation 12
Dokumente, über document()-Funktion laden 68
–dup (Query-Befehl) 100

E

–edit (Query-Befehl) 101
Eingebettete Formulare 137
Elektronische Dokumentation 12
Elemente
Datenbankfehlercodes 32
In FMPXMLLAYOUT-Grammatik 38
In FMPXMLRESULT-Grammatik 37
In fmresultset-Grammatik 33
E-Mail-Nachrichten
Anfängliche Standardkodierungseinstellung 64
Erweiterungsfunktionen für 74
–encoding (Query-Parameter) 103
Entwicklungsmodus, Web Publishing Engine 120
Erneut anmelden, Script 22
<error code-> und <ERRORCODE>-Elemente 113
Erstellen eines neuen Datensatzes 102
Erweitertes Zugriffsrecht für Custom Web Publishing 21
Erweiterungsfunktionen für FileMaker XSLT 58
Siehe auch fmxslt-Erweiterungsfunktionen
Exportieren von XML-Daten 25
Extensible Markup Language (XML) *siehe* XML
Extensible Stylesheet Language Transformations (XSLT) *siehe* XSLT

F

Fehler
Datenbankfehlercode-Elemente 32
Datenbankfehlercodes 113
Erläuterung von Fehlercodes 113
Erweiterungsfunktionen, Fehlercodenummern 121
Fehlerstatus von Erweiterungsfunktionen prüfen 85, 121
pe_application_log.txt (Protokolldatei) 93
pe_server_error.html (Fehlerseite) 121
Protokolldateien für Web-Server 92
Web Publishing Engine-Fehlercodenummern 120
Fehlerbehebung
Custom Web Publishing-Websites 91
XML-Dokumentzugriff 44
XSLT-Stylesheets 55
Feldname (Query-Parameter, nicht für Medienfeld) 104
Feldnamen, voll qualifizierte Syntax 97
–field (Query-Parameter, für Medienfeld) 104
–fieldname.op (Query-Parameter) 105
FileMaker CDML Converter *siehe* CDML Converter
FileMaker Pro, Unterschied zu Web Publishing Engine 25
FileMaker Server, Dokumentationsinformation 12
FileMaker Site Assistant *siehe* Site Assistant

FileMaker-spezifische XSLT-Parameter 66
Filtern von Daten mit Stylesheets 45
–find (Query-Befehl) 101
–findall (Query-Befehl) 101
–findany (Query-Befehl) 101
FMPDSORESULT-Grammatik
  Im Vergleich mit anderen Grammatiken 31
FMPXMLLAYOUT-Grammatik 25, 38–39
  Im Vergleich mit anderen Grammatiken 31
FMPXMLRESULT-Grammatik 25, 36–37
  Im Vergleich mit anderen Grammatiken 31
fmresultset-Grammatik 25, 33–35
  Im Vergleich mit anderen Grammatiken 31
fmxml (Schlüsselwort zur Aktivierung von XML-Publishing) 21, 27
fmxml (Schlüsselwort zur Aktivierung von XSLT-Publishing) 21, 47
fmxslt-Erweiterungsfunktionen
  fmxslt:break_encode(), Funktion 78
  fmxslt:check_error_status(), Funktion 85, 121
  fmxslt:compare_date(), Funktion 82
  fmxslt:compare_datetime(), Funktion 83
  fmxslt:compare_day(), Funktion 82
  fmxslt:compare_time(), Funktion 82
  fmxslt:contains_checkbox_value(), Funktion 80
  fmxslt:convert_datetime(), Funktion 83
  fmxslt:create_session(), Funktion 72
  fmxslt:get_cookie(), Funktion 76
  fmxslt:get_cookies(), Funktion 76
  fmxslt:get_date(), Funktion 81
  fmxslt:get_datetime(), Funktion 82
  fmxslt:get_day(), Funktion 81
  fmxslt:get_fm_date_format(), Funktion 81
  fmxslt:get_fm_date_format()-Funktion 81
  fmxslt:get_header(), Funktion 76
  fmxslt:get_long_date_format(), Funktion 81
  fmxslt:get_long_day_format(), Funktion 82
  fmxslt:get_long_time_format(), Funktion 82
  fmxslt:get_session_object(), Funktion 72
  fmxslt:get_short_date_format(), Funktion 81
  fmxslt:get_short_day_format(), Funktion 82
  fmxslt:get_short_time_format(), Funktion 82
  fmxslt:get_time(), Funktion 81
  fmxslt:html_encode(), Funktion 78
  fmxslt:invalidate_session(), Funktion 72, 73
  fmxslt:regex_contains(), Funktion 79
  fmxslt:remove_session_object(), Funktion 72
  fmxslt:send_email(), Funktionen 74
  fmxslt:session_encode_url(), Funktion 71, 72
  fmxslt:session_exists(), Funktion 72
  fmxslt:set_cookie(), Funktion 76
  fmxslt:set_header(), Funktion 76
  fmxslt:set_session_object(), Funktion 72
  fmxslt:set_session_timeout (), Funktion 72
  fmxslt:set_status_code(), Funktion 76
  fmxslt:url_decode(), Funktion 79
  fmxslt:url_encode(), Funktion 78
Formatieren von Daten mit Stylesheets 45
Formatstrings, Datum und Zeit 83
Formulare, verschachtelt 137

G

Gastkonto
  Aktivieren 22
  Deaktivieren 22
  Mit Custom Web Publishing 22
Generieren einer statischen Seite 64
get_cookie(), Erweiterungsfunktion 76
  CDML-Konvertierung 142
get_cookies(), Erweiterungsfunktion 76
get_date(), Erweiterungsfunktion 81
get_datetime(), Erweiterungsfunktion 82
get_day(), Erweiterungsfunktion 81
get_fm_date_format(), Erweiterungsfunktion 81
get_fm_time_format(), Erweiterungsfunktion 81
get_header(), Erweiterungsfunktion 76
get_long_date_format(), Erweiterungsfunktion 81
get_long_day_format(), Erweiterungsfunktion 82
get_long_time_format(), Erweiterungsfunktion 82
get_session_object(), Erweiterungsfunktion 72
get_short_date_format(), Erweiterungsfunktion 81
get_short_day_format(), Erweiterungsfunktion 82
get_short_time_format(), Erweiterungsfunktion 82
get_time(), Erweiterungsfunktion 81
GIF-Dateien, im Web veröffentlichen 24
–grammar (Query-Parameter) 60, 106
Grammatik, für XSLT empfohlen 60
Grammatiken für XML, Beschreibung 31

H

Header-Funktionen, verwenden 76
Hinweise zur Dokumentation 12, 20
HTML
  Formulare für XML-Abfragen 27
  XML-Daten umformatieren in 25
html_encode(), Erweiterungsfunktion 78

I

Importieren von XML-Daten 25
Informationen zwischen Stylesheets übergeben 65
Inhaltspufferung, verwenden 70
Instant Web Publishing 11
Integrieren von Daten mit Stylesheets 45
invalidate_session(), Erweiterungsfunktion 72, 73
ISO-2022-JP-Kodierung 63
ISO-8859-15-Kodierung 63
ISO-8859-1-Kodierung 63

J

JavaScript
  Definieren von Erweiterungsfunktionen 86
JPEG-Dateien, im Web veröffentlichen 24
jsessionid-Parameter 71

K

Kodierung
  Aufträge 63
  Ausgabe über Element <xsl:output> 64
  –encoding (Query-Parameter) 63, 103
  Erweiterungsfunktionen zur Stringmanipulation verwenden 78
  URL 31, 71
  XML-Daten 32, 40
  XSLT-Stylesheets 64
Konten und Zugriffsrechte
  Für Custom Web Publishing aktivieren 21
  Gastkonto 22
  Scripts 17
Kontrollkästchen, Werte prüfen in 80
Konvertierungsprotokoll, CDML 53

L

Laden zusätzlicher Dokumente 68
–lay (Query-Parameter) 42, 106
–lay.response (Query-Parameter) 42, 106
Layout bei Abfrage von XML-Daten angeben 42
Layoutinformation, in Stylesheet verwenden 69
Layoutinformationen abrufen 103
Layoutnamen abrufen 102
–layoutnames (Query-Befehl) 102
Layouts, für eine XML-Antwort wechseln 42
Logs (Ordner) 93
–lop (Query-Parameter) 107

M

Mail-Nachrichten *siehe* E-Mail-Nachrichten
–max (Query-Parameter) 107
Medienfelder
  Inhalt veröffentlichen 23, 55
  URL-Syntax für Zugriff in XML-Lösungen 30
  URL-Syntax für Zugriff in XSLT-Lösungen 59
  Wie Web-Benutzer auf Daten zugreifen 24
Element <metadata> 34
Metadaten, mit Stylesheets ausblenden 45
method-Attribut, <xsl:output>-Element 64
Migrieren von Web Publishing-Lösungen 19
MIME (Multipurpose Internet Mail Extensions-)-Typen 23
–modid (Query-Parameter) 107

N

Namensräume für
  XML 32
Namespaces für
  XSLT 61
Neue Funktionen in Custom Web Publishing 16
–new (Query-Befehl) 102
Nummern für
  Datenbankfehlercodes 113
  Erweiterungsfunktions-Fehlercodes 121
  Web Publishing Engine-Fehlercodes 120

O

Online-Dokumentation 12
Operatoren, Vergleich 105

P

Parameter für XSLT, FileMaker-spezifisch 66
Parameter, Query s. Query-Strings
Passwort ändern (Script) 22
Passwörter
  Authentifizierung Web-Benutzer 21, 67
  Für Custom Web Publishing definieren 21
  Kein Anmeldepasswort 22
  Passwort ändern (Script) 22
  Zugriff auf XML-Dokumente 69
PDFs 12
pe_application_log.txt (Protokolldatei) 93
pe_internal_access_log.txt (Protokolldatei) 94
pe_server_error.html (Fehlerseite) 121
–process (Query-Befehl) 64, 102

Produktionsmodus, Web Publishing Engine 121
Programmprotokoll 85, 93
Protokolldateien 91, 94
  Beschreibung 92
  pe_application_log.txt 93
  pe_internal_access_log.txt 94
  Protokollierung über <xsl:message>-Element 85
  web_server_module_log.txt 93
  Web-Server-Zugriff 92
  <xsl:message>-Element 93
Pufferung, in Stylesheet verwenden 70

Q

Query-Information, Zugriff in Abfrage 66
Query-String-Referenz 97
Query-Strings 60, 95
  Befehle und Parameter 60, 95
  Befehle, Parameter 40
  Datensätze in Ausschnitte einfügen 98
  Datensätze in Ausschnitten bearbeiten 99
  Query-String-Referenz 97
  Richtlinien für 96
  Statisch definiert in XSLT-Stylesheets 62
  Variablenfeld, Syntax 99
  Veraltete Abfragenamen und Parameter 96
  Voll qualifizierter Feldname, Syntax 97
  XML 40
  XML-Daten abfragen 95
  XSLT-Stylesheets, verwenden in 60
QuickTime-Filme, im Web veröffentlichen 24

R

–recid (Query-Parameter) 107
regex_contains(), Erweiterungsfunktion 79
Reguläre Perl-Ausdrücke, Vergleich von Strings 79
Reihenfolge der XML-Abfrageverarbeitung 43
remove_session_object(), Erweiterungsfunktion 72
Element <resultset> 34

S

Scalable Vector Graphics (SVG), Umwandeln von XML-Daten in 25
Schlüsselwörter zur Aktivierung von Custom Web Publishing 21, 27, 47
Schützen veröffentlichter Datenbanken 22
–script (Query-Parameter) 108
–script.param (Query-Parameter) 108
–script.prefind (Query-Parameter) 108
–script.prefind.param (Query-Parameter) 109
–script.presort (Query-Parameter) 109
–script.presort.param (Query-Parameter) 109
–scriptnames (Query-Befehl) 102
Scripts
  Datenbanksitzungen, aktivieren 73
  Erneut anmelden 22
  Für XML-Abfragen 27
  In Custom Web Publishing 17
  Konten und Zugriffsrechte 17
  Passwort ändern 22
  Tipps und Überlegungen 17
send_email(), Erweiterungsfunktionen 74
Server-seitige XSLT-Stylesheets 45, 57
session_encode_url(), Erweiterungsfunktion 71
session_exists(), Erweiterungsfunktion 72
set_cookie(), Erweiterungsfunktion 76
set_header(), Erweiterungsfunktion 76
set_session_object(), Erweiterungsfunktion 72
set_session_timeout (), Erweiterungsfunktion 72
set_status_code(), Erweiterungsfunktion 76
Shift_JIS-Kodierung 63
Sicherheit
  FileMaker Sicherheitshandbuch 12
  Konten und Passwörter 23
  Richtlinien zum Schützen veröffentlichter Datenbanken 22
  Statisch definierte Query-Strings, verwenden 62
  Zugriff von IP-Adressen beschränken 23
Site Assistant
  Beschreibung 14, 48
  Generierte Stylesheets, Beschreibung 50
  Installieren 48
  Starten 49
  Verwenden 49
  Verwendung vorbereiten 48
Sitzungserweiterungsfunktionen, in Stylesheets verwenden 71, 72
–skip (Query-Parameter) 110
–sortfield (Query-Parameter) 110
–sortorder (Query-Parameter) 110
Speichern von Informationen in Sitzungen 71, 72
SSL (Secure Sockets Layer)-Verschlüsselung 23
Statisch definierte Query-Strings in XSLT-Stylesheets 62
Status, in Sitzungen speichern 71, 72

Strings
Erweiterungsfunktionen zur Stringmanipulation verwenden 78
Vergleich über reguläre Perl-Ausdrücke 79
–stylehref (Query-Parameter) 111
Stylesheets
Anweisung zur Verarbeitung von XML-Stylesheets 43
Beispiele für Verwendung 45
Client-seitig 43
Cookie-Erweiterungsfunktionen 76
Datums- und Zeitformatstrings 83
Datums-, Zeit- und Tages-Erweiterungsfunktionen 81
E-Mail-Nachrichten, senden 74
Entwickeln 57
Erläuterung 45
Erweiterungsfunktionen zur Stringmanipulation 78
Fehlerstatus von Erweiterungsfunktionen, prüfen 85
–grammar-Parameter für 60
Header-Funktionen, verwenden 76
In Website oder Programm verwenden 54
Inhaltspufferung, verwenden 70
Kodierung von 64
Kontrollkästchen, Werte prüfen in 80
Layoutinformation verwenden in 69
Mit CDML Converter erstellen 50
Mit Site Assistant erstellen 48
Query-Strings für 60
Richtlinien für Entwicklung 57
Server-seitig 45, 57
Sitzungsfunktion, verwenden 71, 72
Testen 91
Vergleich von Strings über reguläre Perl-Ausdrücke 79
XSLT, Beschreibung 45
–styletype (Query-Parameter) 112

T

Tages-Erweiterungsfunktionen, verwenden 81
Testen
Websites 91
XML-Ausgabe 91
Textkodierung
Anfängliche Standardeinstellungen 64
–encoding (Query-Parameter) 63, 103
Erweiterungsfunktionen zur Stringmanipulation verwenden 78
Für XSLT-Abfragen 63
Generierte XML-Daten 32
Kodierungseinstellungen 63
Standard für Abfragen und Ausgabeseiten 63
URL 31, 71
–token (Query-Parameter) 65, 112
Tools für XSLT, Beschreibung 14, 48, 50

U

Überblick über Schritte für
XML-Datenzugriff 27
XSLT Publishing 47
Überwachen von Websites 92
Umwandeln von Daten mit Stylesheets 45
Unicode-Zeichen 40
url_decode(), Erweiterungsfunktion 79
url_encode(), Erweiterungsfunktion 78
URL-Syntax für
Medienobjekte in XML-Lösungen 30
Medienobjekte in XSLT-Lösungen 59
XML-Abfragen 29
XSLT-Stylesheets 58
URL-Textkodierung 31
US-ASCII-Kodierung 63
User-Agent-Header, prüfen 65
UTF-8 (Unicode Transformation 8 Bit)
Format 31, 40
Kodierungseinstellung 63

V

Variablenfelder
Datenbanksitzungen, aktivieren 73, 100
Mit Sitzungen verwenden 73, 100
Syntax 99
vCards, Umformatieren von XML-Daten in 25
Veraltete Query-Abfragenamen und Parameter 96
Verarbeiten von XSLT-Stylesheets 102
Verfügbare Datenbanklayouts 102
Verfügbare Scripts 102
Vergleich von Strings 79
Vergleichsoperatoren für Felder 105
Veröffentlichen im Web
Datenbankfehlercodes 113
Direktes Web Publishing verwenden 11
Medienfeldobjekte 23, 55
Mit XSLT 14, 47, 57
mit XSLT 11
QuickTime-Filme 24
Schützen von Datenbanken 22
Verbindung zum Internet oder Intranet 15
Voraussetzungen für das 15
XML verwenden 11, 14, 27
Verschachtelte Formulare 137
–view (Query-Befehl) 103
Voll qualifizierter Feldname, Syntax 97
Voraussetzungen für Custom Web Publishing 15

W

Web Publishing Core
Illustration 26, 46
Interne Zugriffsprotokolle 94
Web Publishing Engine
Administrationskonsole 20–27, 47
Beschreibung 11
Entwicklungsmodus 120
Erzeugte Fehlercodes 113
Produktionsmodus 121
Programmprotokoll 93
Seiten aus XSLT-Stylesheet generieren 46
Vorteile 13
XML-Daten generieren 26
XML-Dokumente erzeugen 27
web_server_module_log.txt (Protokolldatei) 93
Web-Benutzer
Medienfelddaten verwenden 24
Voraussetzungen für Zugriff auf Custom Web Publishing-Lösungen 15
Zugriff auf geschützte Datenbanken 21, 67, 69
Webbrowser
Ausgabe empfangen 11
Rolle in XML-Abfragen 26
Rolle in XSLT-CWP-Abfragen 46
Web-Ordner, Medienfeldobjekte kopieren 24
Web-Server
MIME-Typenunterstützung 23
Protokolldateien 92
Rolle in XML-Abfragen 26
Rolle in XSLT-CWP-Abfragen 46
Web-Sicherheitsdatenbanken, nicht unterstützt 19
Websites
FileMaker Support-Seiten 12
Mit FileMaker Web Publishing Engine erstellen 13
Testen 91
Überwachen 92
Wechseln des Layouts für eine XML-Antwort 42
Werte, prüfen in Kontrollkästchen 80
wpc_access_log.txt (Datei) 94

X

XML
Anweisung zur Verarbeitung von XML-Stylesheets 43
Beschreibung 25
Client-seitige Stylesheets verwenden 43
Daten abfragen 27
Daten filtern 25
Document Type Definitions (DTDs) 33, 36
Fehlerbehebung beim Zugriff auf XML-Dokumente 44
FMPXMLLAYOUT-Grammatik 38
FMPXMLRESULT-Grammatik 37
fmresultset-Grammatik 33
Element <datasource> 33
Element <metadata> 34
Element <resultset> 34
Grammatiken, Beschreibung 31
Im UTF-8-Format kodiert 32, 40
In Datenbank aktivieren 21
Namensräume für 32
Parser 27, 40
Query-Strings 40, 95
Reihenfolge der Abfrageverarbeitung 43
URL-Textkodierung 31
XML 1.0-Spezifikation 25
XML-Daten aus Abfragen generieren 26
Zusammenfassung der Schritte für Zugriff auf XML-Daten 27

XML-Abfrage
Layout angeben 42
XML-Antwort
Layout wechseln 42
XPath-Anweisungen 65
<xsl:stylesheet>-Element 61, 65, 67, 91, 126
<xsl:message>-Element 85
<xsl:output>-Element 64
<xsl:param name="authenticated-xml-base-uri"/>-Parameter 68
<xsl:param name="client-ip"/>-Parameter 67
<xsl:param name="client-password"/>-Parameter 67
<xsl:param name="client-user-name"/>-Parameter 67
<xsl:param name="request-query"/>-Parameter 66
<xsl:param name="xml-base-uri"/>-Parameter 68
<xsl:param>-Element 66
<xsl:template>-Element 67, 68, 91, 92, 126, 129
<xsl:variable>-Element 68
XSLT
Beispiele für Stylesheets 45
Beschreibung 45
CDML Converter, verwenden 50
Cookie-Erweiterungsfunktionen 76
Datums- und Zeitformatstrings 83
Datums-, Zeit- und Tages-Erweiterungsfunktionen 81
E-Mail-Nachrichten, senden 74
Entwickeln von XSLT-Stylesheets 57
Erweiterungsfunktionen für FileMaker 65
Erweiterungsfunktionen zur Stringmanipulation 78
Erweiterungsfunktionen, Referenz 58
Fehlerbehebung in Stylesheets 55
Fehlerstatus von Erweiterungsfunktionen, prüfen 85
FileMaker-spezifische XSLT-Parameter 66
–grammar-Parameter 60
Header-Funktionen, verwenden 76
In Datenbank aktivieren 21
Inhaltspufferung, verwenden 70
JavaScript-Erweiterungen 86
Kontrollkästchen, Werte prüfen in 80
Layoutinformation, verwenden 69
Namespaces für 61
Query-String-Referenz 97
Query-Strings für 60
Seiten aus XSLT-Stylesheet generieren 46
Server-seitige Stylesheets 45, 57
Site Assistant, verwenden 48
Stylesheets in Website oder Programm verwenden 54
Vergleich von Strings über reguläre Perl-Ausdrücke 79
XSLT 1.0-Spezifikation 45
XSLT-CWP-Abfragen 46
xslt-template-files (Ordner) 47, 54, 69
Zusammenfassung der Schritte für Publishing 47
<?xslt-cwp-buffer buffer-content="true"?> (Verarbeitungsanweisung) 70
<?xslt-cwp-query?> (Verarbeitungsanweisung) 58, 62
xslt-template-files (Ordner) 47, 54, 69

Z

Zeit-Erweiterungsfunktionen, verwenden 81
Zeitformatstrings 83
Zugriffsprotokolldateien für Web-Server, Beschreibung 92
Zugriffsrechte 22
Zusammenfassung der Schritte für
XML-Datenzugriff 27
XSLT Publishing 47